Minimal Definition Signatures: Computation and Application to Ontology Alignment

Thesis submitted in accordance with the requirements of
the University of Liverpool for the degree of Doctor in Philosophy by

David Geleta

June 2018

Contents

List of Figures

List of Tables

Notations

The following notations and abbreviations are found throughout this thesis:

$\mathcal{A}$	ABox
$\mathcal{R}$	RBox
$\mathcal{T}$	TBox
$\mathsf{N}_{\mathcal{C}}$, $\mathsf{N}_{\mathcal{R}}$, $\mathsf{N}_{\mathcal{I}}$	concept, role and individual names of $\mathcal{O}$
$\mathcal{O}$	ontology
$\mathcal{M}$	module
$\mathcal{M}^{\mathsf{C}} = \mathsf{Mod}(\{C\}, \mathcal{T})$	module of concept C in TBox $\mathcal{T}$
$\mathcal{J}$	justification
$\mathcal{P}$	power set
$\mathsf{Sig}(\mathcal{T})$	signature (of a TBox)
$\mathsf{Ax}(\mathcal{O})$	set of axioms (of an ontology)
$M = \{\Sigma_1, \ldots, \Sigma_n\}$	set of MDSs
$\mathbf{M} = \{m_1 : \langle \mathsf{e}_i, \Sigma \rangle, \ldots, m_n\}$	set of MDSs
Σ	(minimal) defintion signature
$\Sigma_1^{\mathsf{A}_1}, \Sigma_2^{\mathsf{A}_1}$	two MDSs of the concept $\mathsf{A_1}$
$\mathbf{R}$	removable (signature) entities
$\mathcal{S}, \mathcal{K}, \mathcal{W}$	entity sets
$\mathbf{C}$	cover set
$\mathbf{S}$	collection of subsets of $\mathbf{U}$
$\mathbf{U}$	universe
$\mathbf{F}$	a set of functional dependencies (FDs)
$\mathbf{F}^+$	the closure of $\mathbf{F}$
$\mathfrak{C}$	cover set (set of entities)
$\mathfrak{K}$	set of minimal cover sets
$\mathfrak{R}$	restricted signature (set of entities)
$\mathfrak{S}$	task signature (set of entities)
$\mathfrak{C}^+, \mathfrak{R}^+, \mathfrak{S}^+$	signature closures
$\mathfrak{m} : (\Sigma \to \mathsf{e})$	fMDS

$v(\mathfrak{m})$	fMDS value function
$c(\mathfrak{m})$	fMDS value cost function
$\mathfrak{M} = \{\mathfrak{m}_1 : (\mathsf{A}, \mathsf{B} \rightarrow \mathsf{C}), ..., \mathfrak{m}_n\}$	set of fMDSs
$\mathfrak{M}^+$	closure of an fMDS set
A	alignment
c	correspondence (mapping), simple or complex
r	semantic relation ($r \in \{\equiv, \sqsubseteq, \sqsupseteq, \perp\}$)
A_{GS}	gold-standard alignment
A_1	the first alignment in a sequence
$A_{1,2}$	alignment between $\mathcal{O}_1$ and $\mathcal{O}_2$
A^+	alignment closure
R	reference alignment
pr	precision
rc	recall
Λ	set of alignments
Ω	set of ontologies
DL	Description Logic
MAS	Multi-Agent Systems
OAEI	Ontology Alignment Evaluation Initiative
OWL	Web Ontology Language
W3C	World Wide Web Consortium
DS	Definition Signature
MDS	Minimal Definition Signature
CDS	Concept Definition Signature
RDS	Role Definition Signature
DP	Definition Pattern

Preface

This thesis is primarily my own work. The sources of other materials are identifed.

Abstract

In computer science, ontologies define a domain to facilitate knowledge representation and sharing, in a machine processable way. Ontologies approximate an actual world representation, and thus ontologies will differ for many reasons. Therefore knowledge sharing, and in general semantic interoperability, is inherently hindered or even precluded between heterogenous ontologies. Ontology matching addresses this fundamental issue by producing alignments, i.e. sets of correspondences that describe relations between semantically related entities of different ontologies. However, alignments are typically incomplete. In order to support and improve ontology alignment, and semantic interoperability in general, this thesis exploits the notion of implicit definability. Implicit definability is a semantic property of ontologies, signatures, and concepts (and roles) stating that whenever the signature is fixed under a given ontology then the definition of a particular concept (or role) is also fixed.

This thesis introduces the notion of minimal definition signature (MDS) from which a given entity is implicitly definable, and presents a novel approach that provides an efficient way to compute in practice all MDSs of the definable entities. Furthermore, it investigates the application of MDSs in the context of alignment generation, evaluation, and negotiation (whereby agents cooperatively establish a mutually acceptable alignment to support opportunistic communication within open environments). As implicit definability permits defined entities to be removed without semantic loss, this thesis argues, that if the meaning of the defined entity is wholly fixed by the terms of its definition, only the terms in the definition are required to be mapped in order to map the defined entity itself; thus implicit definability entails a new type of definability-based correspondence correspondence. Therefore this thesis defines and explores the properties of definability-based correspondences, and extends several ontology alignment evaluation metrics in order to accommodate their assessment. As task signature coverage is a prerequisite of many knowledge-based tasks (e.g. service invocation), a definability-based, efficient approximation approach to obtaining minimal signature cover sets is presented. Moreover, this thesis outlines a specific alignment negotiation approach and shows that by considering definability, agents are better equipped to: (*i*) determine whether an alignment provides the necessary coverage to achieve a particular task (align the whole ontology, formulate a message or query); (*ii*) adhere to privacy and confidentiality constraints; and (*iii*) minimalise the cardinality of the resulting mutual alignment.

Acknowledgements

I would like to thank my supervisors, Dr Terry R. Payne, Dr Valentina Tamma, and Prof Frank Wolter for their continuous support, encouragement and patience during this challenging, but rewarding 'journey', as well as for giving me the opportunity to undertake the PhD in the first place. Furthermore, I would also like to express my gratitude to my advisors, Prof Wiebe Van der Hoek, Prof Simon Parsons, and Dr Ullrich Hustadt for their feedback and advice; the KR research group (especially Dr Fabio Papacchini) for their invaluable comments.

To Richard M. Williams and Eric Schneider, 'I will say this': gentlemen, it was my sincere pleasure to have gone through this experience together. We have: had the most amazing coffee-times and philosophized about the finer things in life; fought dragons; enjoyed the best culinary experience that the immediate proximity of the university has to offer; but above all, acted as a support group through thick and thin. I also thank the rest of the fine people of the famed office of 2.13, Matoula Kotsialou, Jeffrey Rafael, Gabrielle Dos Santos, et al. Last but certainly not least, I would like to thank my family, especially my mother, Agnes (K*öszönö*m Anyukám, which in Hungarian translates to 'Thanks, Mom'); my 'brother from another mother': Bandi; and my amazing wife, Jacqui.

Part I

Background and Context

Chapter 1

Introduction

"[Robinson: I] made it my business to teach him everything that was proper to make him useful, handy, and helpful; but especially to make him speak, and understand me when I spoke; and he was the aptest scholar that ever was; and . . . so pleased when he could but understand me, or make me understand him, that it was very pleasant for me to talk to him. Now my life began to be so easy . . . , I cared not if I was never to remove from the place where I lived."– Daniel Defoe, *The Life and Adventures of Robinson Crusoe* (1719)

In Daniel Defoe's famous historical fiction novel, Robinson, an Englishman from the town of York, after twenty-four years of seemingly endless solitude spent as a castaway on a remote tropical island, rescues a native man from a local cannibal tribe [32]. Starting with gestures and basic words, Robinson eventually teaches proper English to his companion, thus allowing them carry out meaningful communication and cooperation. As an effective team, they survive the 'Island of Despair' and ultimately depart to England. In open dynamic environments, such as in the real world, or in virtual environments (Multi-Agent Systems [151], or MAS), the ability of gaining understanding over heterogeneous knowledge is integral to coalition formation and semantic interoperability. This thesis provides new insights into this process, by exploiting the notion that the same knowledge can be articulated (i.e. defined) in different ways, and thus by identifying the signatures of such definitions, the chance of carrying out meaningful communication based on an incomplete common vocabulary increases. This chapter is organised as follows: Section 1.1 introduces context and motivation; Section 1.2 describes the notion of definability, while Section 1.3 presents a synopsis of the main method, which exploits definability. Research questions and contributions are presented in Section 1.4; and the structure of this thesis is outlined in Section 1.5.

1.1 Background and Motivation

In computer science and information technology, ontologies are machine processable artefacts that capture *knowledge* about a domain of interest [68]. An ontology provides a model of such a domain by introducing a vocabulary and a specification of the meaning

of terms used in the vocabulary. Ontologies are typically encoded in a formal knowledge representation language that is characterised by well-defined syntax and formal semantics. This permits the machine-processing of ontologies; furthermore, it provides the ability to precisely describe the meaning of an ontology vocabulary, thus for example, facilitating automated query answering (or reasoning) about the modelled knowledge. The family of *Description Logics* (DLs) is one of the most prominent knowledge representation languages used for representing ontological axioms [5]. A DL ontology describes a domain in terms of concepts (also called classes), roles (or properties) and individuals (or instances). A concept denotes a set of objects, e.g. Mother; a role denotes a relation between objects (i.e. a set of object pairs), e.g. hasChild; and an individual denotes a particular object, e.g.Man:*Abe*, in an interpretation. These three basic building blocks are collectively referred to as *entities*. A set of entities is also referred to as a *signature*. Entities and logical operators, such as concept and role constructors, form axioms; for example, the logic formula $\mathsf{Mother} \equiv \mathsf{Women} \sqcap \exists \mathsf{hasChild}.\top$ states that mothers are those women that have children. DLs provide the logical foundation of the widely used Web Ontology Language (OWL), the World Wide Web Consortium's (W3C[1]) standard ontology language [66].

In addition to knowledge representation and reasoning, the main aim of ontologies is to facilitate *sharing and integration* between knowledge-based systems [68]. However in open, dynamic environments such as the Semantic Web [11], distinct systems cannot be assumed to adhere to the same ontologies, even when representing the same domain. Therefore knowledge sharing, and in general semantic interoperability is inherently hindered or even precluded between distinct ontologies, as independently designed ontologies typically introduce syntactically and semantically different domain conceptualisations. Ontologies approximate actual world representations and thus will differ due to the different requirements they adhere to, the assumptions they make, and contexts they are used in. *Ontology matching* addresses this fundamental and ubiquitous issue (i.e. ontology *heterogeneity*) by producing *alignments*, i.e. sets of *correspondences* that describe the logical relations (i.e. $\equiv, \not\equiv, \sqsubseteq, \sqsupseteq, \perp$ etc.) between semantically related entities of different ontologies [106]. For example, one ontology may define a large vehicle for transporting goods as a Lorry, while another may refer to the same concept as a Truck. In order to reconcile this terminological heterogeneity, ideal matching systems would produce the correspondence $\langle \mathsf{Truck}, \mathsf{Lorry}, \equiv \rangle$, which describes the two concepts as interchangeable synonym words under the ontologies. Ontology matching (also called *alignment*) has become an established research field with several growing sub-fields, and received increasing interest in the past two decades [47, 106, 124]. In a recent and comprehensive literature review, Otero-Cerdeira et al. [106] reported that between 2003 and 2013, 694[2] articles were published in the following areas: reviews, matching techniques

[1]The W3C, founded by Tim Berners-Lee, is an international community that develops standards for the Web [147].

[2]This number represents only those publications where ontology matching was the main focus. The total number of papers, related to ontology matching, was over 1600.

and systems, practical frameworks and applications, and evaluation. In recent years many diverse ontology matching techniques and systems have emerged, leading to the development of alignment evaluation and the *Ontology Alignment Evaluation Initiative* (OAEI). Every year since 2004, the OAEI organises campaigns to evaluate the efficiency of matching systems, and the quality of alignments [23, 48].

Despite extensive research efforts, the ontology alignment problem is still an open question. Although the ontology matching community has proposed many distinct methods, *no single approach* is necessarily suitable for all matching scenarios [23, 124]. Furthermore, alignments are typically *incomplete*, providing only a partial coverage of an ontology vocabulary. Incompleteness can occur due to the shortcomings of matching approaches, such as producing incorrect, imprecise correspondences. For example, given the correct correspondence (or mapping) $\langle \mathsf{Truck}, \mathsf{Lorry}, \equiv \rangle$, the mapping $\langle \mathsf{Truck}, \mathsf{Lorry}, \bot \rangle$ which describes that the two aligned concepts are disjoint, i.e. semantically unrelated notions, is incorrect; whereas the mapping $\langle \mathsf{Truck}, \mathsf{Lorry}, \sqsubseteq \rangle$ is not precise, as it only states that every Truck is a Lorry (and not vice versa), and thus does not declare them as synonym concepts. In addition to the previously mentioned terminological heterogeneity problem (which is typically a non-trivial but, to some degree, solvable challenge for many current matching systems), there are several heterogeneity types that further increase the complexity of ontology matching. Conceptual or semantic heterogeneity occurs if two ontologies differ in: *coverage* (addressing different areas of the same subject), *granularity* (employing different level of detail), or *scope* (take different perspectives). Resolving semantic heterogeneity often requires matching systems to produce complex correspondences; whilst a (simple) mapping describes a relation between two concept or role names, a complex mapping can be used to describe more sophisticated relations. For example, the correspondence $\langle \mathsf{Mother}, \mathsf{Women} \sqcap \exists \mathsf{hasChild}.\top, \equiv \rangle$ states, that the concept name Mother of one ontology, can be defined as the complex concept $\mathsf{Women} \sqcap \exists \mathsf{hasChild}.\top$ in the other, pairwise aligned ontology. However, in 2008, Stuckenschmidt et al. argued that existing approaches often fail to compute complex correspondences: typically, systems are only able to identify simple equivalence statements between concept or role names, but often fail to identify richer semantic relation between elements of different ontologies [131]. Although several approaches have addressed complex matching [1, 58, 113, 121, 148] since 2008, the argument still holds.

Creating ontology alignment is a significant, but not the only requirement of semantic interoperability in dynamic environments. In order to be able to carry out meaningful communication, knowledge-based *agents* (autonomous systems that typically exploit ontologies to model the world, and their internal preferences, as well as those of other agents) are often required to cooperatively establish a *mutually acceptable alignment*, which is achieved by agents engaging in a negotiation process. This process, known as *ontology alignment negotiation*, has became an established and active research area that is concerned with supporting opportunistic communication within open agent environments [110]. Assuming a set of existing alignments, that are either stored from

previous interactions, or computed on the fly, negotiation is carried out via argumentation [40, 89, 139] or by a dialogue framework [109], whilst adhering to internal preferences, without compromising confidential knowledge, and avoiding alignment-based conservativity violations to occur [83].

1.2 Definability

In order to support and improve ontology alignment, and semantic interoperability in general, this thesis exploits the notion of *implicit definability*. Implicit definability is a semantic property of ontologies, signatures, and concepts (and roles) stating that whenever the signature is fixed under a given ontology then the definition of a particular concept (or role) is also fixed [12]. For example, given an ontology $\mathcal{O} = \{\mathsf{C} \equiv \mathsf{A} \sqcup \mathsf{B}, \mathsf{A} \sqsubseteq \neg\mathsf{B}, \mathsf{D} \sqsubseteq \exists \mathsf{r}.\top\}$, where the concept C is defined explicitly (by the axiom $\mathsf{C} \equiv \mathsf{A} \sqcup \mathsf{B}$), and concept A is defined implicitly under $\mathcal{O}$ by the set of general concept inclusions $\{\mathsf{C} \equiv \mathsf{A} \sqcup \mathsf{B}, \mathsf{A} \sqsubseteq \neg\mathsf{B}\}$, i.e. A is implicitly definable under $\mathcal{O}$ by the definition signature $\Sigma^{\mathsf{A}} = \{\mathsf{B}, \mathsf{C}\}$. Any concept, or role that is not defined (or definable) either explicitly or implicitly, is referred to as an *undefined* entity[3].

Beth definability [12, 72] is a well-known property from classical logic, that relates the notion of *implicit definability* to the one of *explicit definability*, by stating that every implicitly defined concept is also explicitly definable, in any definitorially complete DL language [135]. Thus, in the previous example, A can be explicitly defined by the axiom $\mathsf{A} \equiv \mathsf{C} \sqcap \neg\mathsf{B}$. Definability in general (and Beth definability in particular) has been utilised within DLs to generate syntactically different, albeit semantically equivalent definitions (i.e. for *"rewriting"*). *Rewriting* is primarily used for: (*i*) extracting equivalent terminology from a general TBox [5]; and (*ii*) finding equivalent query rewritings in ontology-based data access scenarios [123].

Ten Cate *et al.* presented a method to determine whether a particular entity is implicitly definable with a given signature for some ontology, and shown that testing implicit definability can be reduced to entailment checking [135]. The computational complexity of determining whether a concept is implicitly definable with a given signature depends on the complexity of the entailment check, which is predicated on the expressivity of the given DL language. In this thesis the entailment check reasoning task of deciding implicit definability is dedicated to an external reasoner system, and considered as constant time single step computation, i.e. treating them as a call to an *oracle* [107] in the number of axioms in the ontology where each oracle call may itself take exponential time in the number of axioms in the ontology. The implicit definability check process works in languages that do not accept Beth definability.

The core of this thesis is focused on obtaining *definition signatures* (DSs), as it is later described in Section 1.3 and in Part III, several ontology engineering and ontology alignment tasks may benefit from their usage. A definition signature is an entity set

[3]The notion of definability does not (directly) extend to assertional knowledge, such as individuals.

from which an entity is implicitly definable under the ontology which fixes the meaning of the definable entity along with its signature entities. The signature of an ontology (i.e. a set which contains all of its constituent entities), is a DS for all of its defined entities. As definition signatures may contain redundant members, their size could be very large (e.g. one version of SNOMED CT contains over 376,000 concepts). Thus, to accommodate the notion of minimality, *minimal definition signatures* (MDSs) are introduced in this thesis. The minimality property of an MDS refers to minimising the size of the signatures, by eliminating superfluous entities. However, a defined entity may have multiple unique MDSs (where the difference of any two MDSs is not an empty set) under an ontology, with the same cardinality. Clearly, every MDS is also a DS, and any DS may contain at least one, but potentially many MDSs.

The potential benefit of using MDSs increases when the complete set of MDSs is known. However, determining the complete set of MDSs of definable concepts and roles is a challenging task, as the number of different minimal definition signatures of a given defined concept or role is potentially equivalent to the size of the power set of the ontology signature (excluding the definable concept or role name itself), where each candidate signature needs to be examined whether it actually implicitly defines a particular entity, as well as ensured that the definition signature is minimal. This is especially problematic for large scale ontologies (containing hundreds of thousands of entities), such as SNOMED CT [36] or FMA [114]. Therefore, reducing the search space is highly desirable.

Building on the implicitly definability check method, this thesis presents a novel, pragmatic approach that provides an efficient[4] way to compute *in practice* all MDSs of defined entities. Furthermore, to assesses the prevalence, extent and merits of definability in existing ontologies and the impact it has in supporting semantic interoperability, and to analyse the practical applicability and behaviour of the proposed algorithms for computing definability, this thesis reports on the results of an empirical analysis performed over a wide range of OWL ontologies. The empirical evaluation has shown that definability computation is feasible for most real world ontologies, and in some cases, it can be useful in dynamic environments due to the fact that (if not the complete set, then typically) a subset of MDSs can be found by algorithms using a polynomial number of calls to the oracle in the number of axioms in the ontology where each oracle call may itself take exponential time in the number of axioms in the ontology. As for the practical applications of this work we only use definition signatures that are determined by using the ontology language independent implicit definability check, we are not interested in explicit definitions. Therefore it was not assessed whether the ontologies used for the empirical evaluation accept the Beth definability property, because it was not a crucial requirement to be able to generate explicit definitions.

It is often difficult for humans to identify the specific axiom set that implies a particular case of definability. An explicit concept definition is always formalised as a single

[4]Excluding the complexity of the implicit definability check, which is delegated to an external oracle, i.e. a reasoner.

axiom, whereas the definition of an implicitly defined concept is derived from an axiom set. Thus, implicit definitions are often not straightforward to recognise and interpret. *Justifications* can be used to validate definability and to provide a set of axioms supporting an entailment. A justification $\mathcal{J}$ for an entailment η in an ontology is the ontological fragment in which η holds (i.e. a set of TBox axioms such that $\mathcal{J} \subseteq \mathcal{O}$) [73]. The implicit definition check works by testing whether a possible entailment holds. This makes it easy to extract the relevant axioms containing the implicit definition by extracting all justifications for that entailment, which is a standard service with off the shelf, highly optimized tooling [73].

As part of the empirical analysis presented later in this thesis, MDSs were computed for numerous ontologies and validated by obtaining their corresponding justifications. By studying the composition of DSs (their cardinality, and the type and number of their member entities) together with their justifications (their size, and the type of their constituent axioms), a number of *definition patterns* were identified. Although for the bigger picture of this thesis the exact definitions are not important it is still interesting to know what kind of explicit definitions can be found for implicitly definable concept and role names. These patterns aim to generalise the frequent forms of creating definitions that occurred in the evaluation corpus. The set of patterns is not exhaustive, i.e. it is not guaranteed to represent all definitions, however, as shown by empirical analysis (Section 5.2.2), it can cover a significant number of cases. In addition to the validation and the interpretation of definability, the identifiable definition patterns permit *heuristic-based definition axiom generation*, i.e. the generation of an explicit definition of a defined entity (according to some inference rule), by processing a given DS and a justification.

Moreover, the empirical analysis has also highlighted how the computation of MDSs can help in identifying modelling errors in ontologies. Three types of errors were formalised, each of which can be automatically detected, although their repairs require the involvement of an ontology engineer and a domain expert.

1.3 Exploiting Definability

The second part of this thesis explores the use of implicit definability, or more precisely, the various aspects of employing minimal definition signatures in semantic interoperability. F. van Harmelen *et al.* have shown, that *knowledge-based tasks* (amongst other parameters) are typically characterised by their signature, thus the prerequisite for any task is that it must be *covered* by the signature which is available for the party who intends to carry out the given task [142]. In order to determine whether a given *task signature* is coverable by an available, *restricted signature* (e.g. an incomplete alignment), each entity of the task signature must be individually examined. An entity is considered to be covered either if it appears in the restricted signature, or if it is implicitly definable using only the restricted signature members thus the entity in question can be removed without semantic loss. Although task coverage is trivial to establish,

determining the smallest (minimal) signature that covers a given task signature poses a challenge, as the complete set of MDSs needs to be known, and all combinations of the MDSs are required to be explored, for each entity in question. Thus this thesis introduces and characterises the exponential time complexity *ontology signature coverage problem*; moreover it proposes and empirically evaluates a novel approach which, by assuming a priori obtained complete set of MDSs, efficiently computes an approximation of the smallest entity combination (i.e. the *minimal signature cover set*) that covers a given task signature. Although this problem is potentially easier then the reasoning task of deciding implicit definability, we note that the latter is dedicated to an oracle (i.e. we do not attempt to tackle that problem as it is out of the scope of this thesis), while the former problem is addressed by this work because in oppose to entailment checking, there is no such system which would provide a solution for minimal signature coverage.

Neither matching systems, nor evaluation measures that assess the quality of produced alignments and the performance of matchers, have considered the notion of implicit definability and the use of MDSs in the context of ontology alignment. As implicit definability permits implicitly defined entities to be removed without semantic loss, this thesis argues, that if the meaning of the defined entity is wholly fixed by the terms of its definition, only the terms in the definition are required to be mapped in order to map the defined entity itself[5]; thus implicit definability entails a new type of correspondence definability-based (or implicit) correspondence, based on the definition signatures of entities in the aligned ontologies and the available alignment. For example, given an ontology $\mathcal{O} = \{\mathsf{C} \equiv \mathsf{A} \sqcup \mathsf{B}, \mathsf{B} \sqsubseteq \neg\mathsf{A}\}$ and the alignment $A = \{\langle \mathsf{C}, \mathsf{C}', \equiv\rangle, \langle \mathsf{B}, \mathsf{B}', \equiv\rangle\}$ that maps $\mathcal{O}$ to $\mathcal{O}'$, the implicitly defined concept A can be removed without semantic loss w.r.t. the alignment, yielding a *definability-based correspondence*, i.e. $\langle \mathsf{A}, \mathsf{C}' \sqcap \neg\mathsf{B}', \equiv\rangle$. Such a typically *complex correspondence* describes a relation between a defined entity (or its description) in one ontology, and a potentially complex concept (or role) in an aligned ontology, based on the definitions of the aligned entity. This thesis suggests that accounting for definability-based correspondences in alignments can:

- increase *alignment coverage* (the ratio of elements of the ontology which are mapped [31]) as otherwise unmapped entities may become covered by considering definability-based correspondences;
- increase *coverage retention*, i.e. removing some mappings from an alignment does not necessarily effect its expressive capacity (i.e. coverage) as some entities may be mapped by a simple, and a definability-based correspondence;
- increase *compactness*, i.e. for a given knowledge-based task signature, only a subset of an alignment may be necessary to provide coverage.

In order to be able to provide accurate metrics that account for definability-based correspondences, the existing evaluation models need to be extended. Ontology alignments

[5] An entity e is considered to be covered, or mapped, by an alignment A if there exists a correspondence c such that $\{c \in A | c : \langle \mathsf{e}, \mathsf{e}', r\rangle\}$, where r denotes the relation between e and another entity e' of a pairwise aligned ontology [31].

are commonly evaluated with the de-facto standard, *precision* and *recall* metrics, where their compliance is assessed with respect to reference alignments (i.e. a gold standard for a particular alignment problem) [23, 47]. These well-understood metrics, originally adopted from the information retrieval field, have been extended over the years to overcome limitations in the way that they evaluate alignments. Euzenat *et al.* introduced *semantic precision and recall* metrics to address the issue that semantically equivalent, but syntactically different alignments are potentially assigned different precision and recall scores [44]. Their approach compares the semantic closure of alignments, i.e. the complete set of correspondences, which is entailed by the union of the ontologies merged by the corresponding alignment, whereas its syntactic variants consider only the explicitly stated correspondences. An argument can be made that if an ontology can express the same knowledge by considering definability-based correspondences, then these should also have the same precision and recall scores. David et. al. introduced the *alignment coverage and path* metric family, which measures ontology similarity in the alignment space (a network of ontologies connected by alignments) by evaluating the similarity between two ontologies with regard to the set of available alignments between them [31]. This alternate approach employs the notion of alignment paths, i.e. sequence of alignments between particular ontologies. In contrast to the use of (both traditional and semantic) precision and recall metrics, an alignment is not evaluated against a reference alignment, but assessed with respect to a given signature (set of concepts and roles), to establish the quality of the provided coverage. These metrics are useful for evaluating alignments when *similarity* must denote the ability to transfer information within a knowledge-base network, e.g. in forming knowledge-based agent coalitions, the ability to effectively communicate has vital importance. As considering definability-based correspondences effects the coverage and the distinguishability of aligned entities, and potentially strengthen existing paths (by providing alternative connections, i.e. mappings), the aforementioned metric should be extended.

The final step to improved semantic interoperability involves employing definability and minimal definition signatures in agent coalition formation, and ontology alignment negotiation [42, 83, 89, 110]. As the complete set of MDSs can either be pre-computed, or a subset of MDSs can be found in reasonable time for many real world ontologies, this thesis argues that MDSs can be employed in dynamic environments. By considering definability-based evaluation metrics, agents are better equipped to determine whether an alignment provides the necessary coverage to achieve a particular task (i.e. align the whole ontology or just those entities necessary to formulate a message or query, etc.); or to select suitable coalition partners. Furthermore, identifying minimal signature coverage can minimise the cardinality of the cooperatively established mutual alignment, thus reducing the efforts (in terms of cost, and time) of the ontology alignment negotiation process. Moreover, alternative definitions can aid agents in adhering to privacy and confidentially constraints, by exposing only the essential part of their ontology during

the negotiation process, or disclosing only a sufficient part of their ontology (i.e. particular subset of the ontology whose alignment permits a particular task) to the alignment systems, in the case where no a-priori alignment exists.

1.4 Research Questions and Contributions

Research Questions. The underlying motivation of this research was to enhance semantic interoperability by exploiting implicit definability and MDSs. The *main hypothesis* is that by obtaining MDSs, semantic interoperability can be supported and improved. The research was guided by the following questions, divided into five themed groups:

1. *Definability*
   (a) What status does an entity assume w.r.t. to definability under an ontology?
   (b) Does the notion of definability apply to roles, and how can role definability be determined? (While this is a simple extension of concept definability, the practical side may require some attention.)
   (c) How can instances of definability be validated and explained in a human comprehensible way?
   (d) What are the frequent forms (patterns) of defining concepts and roles? Can patterns be used to generate definition axioms?
2. *Minimal Defintion Signatures*
   (a) As the number of MDSs of a given defined entity is exponential in the size of the ontology signature, can an efficient (or at least a practical) approach be found which computes all or some MDSs of definable entities.
   (b) What is the prevalence of definability in real world ontologies?
   (c) Can definition patterns and MDSs be used to identify ontology modelling errors, and thus assist the ontology engineering process?
3. *Signature coverage*: given that the number of candidate cover sets is exponential in the size of the ontology, can an efficient approach be found to compute a minimal cover set?
4. *Definability-based correspondences*
   (a) Can a defined entity be removed without semantic loss under an alignment?
   (b) What are the implications of including definability-based correspondences in alignments, how does it effect alignment properties (coverage, consistency, etc.)?
   (c) How can definability-based alignments be evaluated?

5. *Knowledge-based agents*: what are the implications of using minimal definition signatures, and minimal covers in:

   (a) coalition formation?

   (b) alignment negotiation?

Contributions. The research presented here makes the following contributions:

1. The prerequisite of obtaining and exploiting MDSs is identifying whether a given entity is defined under an ontology, thus this thesis provides an *optimised approach to establishing the definability status* of concepts and roles.

2. In addition to introducing the notion of *definition signature minimality*, a novel approach is presented that provides an efficient way to *compute in practice all minimal definition signatures* of the defined entities. Moreover, this thesis assesses the prevalence, extent, and merits of definability over large and diverse corpora, and provides the basis for its use in ontology alignment. Furthermore, an analyses of the behaviour of the proposed definability algorithms is given.

3. As a result of the empirical analysis of MDS computation, a non-exhaustive list of the frequent forms of definition axiom have been identified. The list of such *definition patterns* is presented, complete with justifications designed to aid human interpretation and validation of definability (i.e. how a given entity is defined under an ontology), for a subset of all possible definitions. Furthermore, entity definition patterns serve as input for a novel *heuristic-based rewriting approach*, which produces definition axioms.

4. The analysis has also highlighted how the computation of MDSs can help in *the identification of modelling errors*. This thesis formalises several errors and provides algorithms for the recognition of such errors.

5. In order to better facilitate and optimise semantic interoperability, this thesis:

   (a) introduces and characterises the non-polynomial time complexity *ontology signature coverage problem*;

   (b) proposes and empirically evaluates a novel approach, which efficiently finds *approximations of minimal signature cover sets*;

   (c) presents a new type of, definability-based, correspondence and extends several ontology evaluation metrics in order to facilitate the assessment of such correspondences;

   (d) it empirically tests the hypotheses that definability-based correspondences can potentially increase alignment (*i*) coverage, (*ii*) coverage retention, and (*iii*) compactness.

6. By combining the ability: (*i*) to obtain some MDSs on the fly, or use a pre-computed set; (*ii*) to minimalise the required coverage for ontology signatures, e.g. knowledge-based tasks; and (*iii*) to assess coalition partners suitable for knowledge-based interactions; this thesis sketches a novel, MDS based *agent coalition formation*, and a *ontology negotiation approach* that by building on the state of the art, improves semantic interoperability.

7. Open source implementations of all *software* developed for this thesis, implementing the aforementioned approaches and algorithms were made available[6] in order to support further research in the area.

Published Work[7]. Part of this thesis have been published or are currently under review:

- David Geleta, Terry R Payne, and Valentina Tamma, *Minimal Coverage for Ontology Signatures*, In: 13th OWL: Experiences and Directions Workshop and 5th OWL reasoner evaluation workshop (OWLED – ORE), Springer, 2016, In Review [55].

- David Geleta, Terry R Payne, and Valentina Tamma, *An Investigation of Definability in Ontology Alignment*, In: The 20th International Conference on Knowledge Engineering and Knowledge Management (EKAW), Springer, 2016, To Appear [51].

Other parts of this thesis were made public as technical reports:

- David Geleta, Terry R Payne, and Valentina Tamma, *Computing Minimal Signature Coverage for Description Logic Ontologies,* Technical Report ULCS-16-004, University of Liverpool, 2016 [53].

- David Geleta, Terry R Payne, and Valentina Tamma, *Computing Minimal Definition Signatures in Description Logic Ontologies,* Technical Report ULCS-16-003, University of Liverpool, 2016 [52].

1.5 Thesis Outline

This thesis has been divided into four parts, and comprises nine chapters.

Part I establishes the context and motivation of this thesis, as well as explains the guiding questions, and the contributions of the presented research (Chapter 1). Chapter 2 provides an introduction to the fundamental principles and formalisms used in Description Logics and the Web Ontology Language.

[6] `https://bitbucket.org/dgeleta/owl-definability`
[7] Part II: [51, 52], Chapter 7: [53, 55], Chapter 8: [54]

Part II presents the notion of Minimal Definition Signatures (MDSs) and their computation. Chapter 3 provides the theoretical background of definability and the algorithms for computing definability-status. It then introduces definition patterns that aid the comprehension of definability, and presents a heuristic-based rewriting approach. The optimised MDS computation algorithms are detailed in Chapter 4. Chapter 5 presents an empirical analysis over a wide range of OWL ontology corpora, and assesses the prevalence of definability and the behaviour of the proposed algorithms for computing definability.

Part III explores the various applications of MDSs and their merits. Chapter 7: introduces the *signature coverage problem* and provides an approach to computing approximations of minimal cover sets, as well as an empirical evaluation on its performance. Chapter 6 introduces ontology matching, the properties and evaluation of alignments, introduces ontology alignment negotiation, and surveys notable alignment negotiation approaches; then Chapter 8 investigates definability-based correspondences, and extends alignment evaluation metrics to support such correspondences. An empirical analysis that assesses the new metrics is also provided.

Part IV summarises the main results; outlines possible future work regarding to the approaches presented in thesis, as well as it provides a sketch for a definability-based agent coalition-formation and a negation framework; and concludes the thesis (Chapter 9).

Chapter 2

Ontologies and Description Logics

Ontologies support the sharing of knowledge across domains and applications by providing a common, ideally machine processable vocabulary. This chapter introduces the definitions, terminology and fundamental principles of ontologies (Section 2.1), and describes the foundations of one of the most prominent knowledge representation languages, Description Logics (Section 2.2), which supports the work presented in this thesis. Description Logics provide the logical foundations of OWL, the standard ontology representation language family of the Semantic Web (Section 2.3); OWL ontologies facilitate the evaluation of the approaches that introduced in later chapters.

2.1 Ontologies

In the classical sense, *ontology* is a "*branch of metaphysics concerned with the nature or essence of being or existence*" [111]. The term was originally described by Aristotle (384–322 BC), the ancient Greek philosopher and scientist, in what is considered to be one of the greatest philosophical works: Metaphysics (IV, 1). Aristotle defines ontology as the science of being that tries to answer the questions: what is a being; how could things be characterised; and how should things be classified?

Over the past several decades, ontologies have received increasing interest from the Computer Science research community. Many different views have emerged on the (ironically itself ontological) question of what is an ontology [10, 14, 57, 69, 70]. The most widely cited definition of the meaning of ontologies in the area of Artificial Intelligence was given by Thomas Gruber in 1993 [68]; this thesis conforms to Studer et al. refinement of Gruber's definition, which states that "*an ontology is a formal, explicit specification of a shared conceptualisation*" [132]. In this definition, *"formal"* refers to the requirement for machine-readability and *"explicit"* means that the meaning of ontological terms is precisely defined. Thus an ontology provides a model (or *"specification of a conceptualisation"*) about a *domain of interest* (such as water-polo, medicine, company policy etc.) that is *shared by a group of users* (e.g. shared between humans, machines, or both).

Most ontologies, regardless of the knowledge representation language they are formalised in, provide conceptualisations by using the following common components:

- *Concepts* denote a sets of objects that share common characteristics and properties. For example Mother is a set (or class) which could be interpreted as the set of all women who have children.

- *Relations* describe relationships between either concepts or individuals. For example, the relation hasChild could be used to associate parents with their child.

- *Individuals* represent actual instances, or objects; e.g. the sentence Mother($Cersei$) states that the individual named $Cersei$ is an instance of the concept Woman (i.e. Cersei is a woman); whereas the statement hasChild($Cersei, Tommen$) declares that $Cersei$ has a child named $Tommen$.

An ontological model introduces a *common vocabulary* that provides names for classes (concepts), properties of classes, as well as it defines relationships between classes and properties.

Ontologies are used in a wide range of application areas and context, thus there are many different types presented in the literature. Van Heisjt provides the following classification according to the *level of generality* an ontology employs in its conceptualisation and the *type of knowledge* an ontology describes [143]:

- *Domain* ontologies (or domain-specific ontologies) provide models about a specific area of interest, such as conference organisation, academia or water-polo.

- *Task* ontologies describe generic or domain-specific activities to support the reuse of problem solving knowledge, e.g. repairing a PC, diagnosing a patient etc.

- *Application* ontologies are created to serve a specific purpose (application), and are often built by using particular domain and a task ontologies.

- *Upper-level/Generic* (also known as foundation) ontologies conceptualise very notions that are contextually independent of the domain they are used in, e.g. time, space, algebra etc.

Ontologies may also be classified by assessing the *level of formality* used in their modelling. Uschold and Gruninger identify four categories of formality [141]:

- *Highly informal* ontologies describe terms using natural language, which is highly ambiguous due to lacking semantics.

- *Semi-informal* ontologies also use natural language, however, ambiguity is reduced (but not necessarily eliminated) by imposing restrictions and structure.

- *Semi-formal* ontologies use artificial languages (without formal semantics) to describe a given domain of interest.

- *Highly-formal* ontologies are expressed using some knowledge representation formalism, with well-defined syntax and formal semantics, e.g. Description Logics.

McGuinness introduced the idea of an "*ontology spectrum*", which provides ontology classification based on the detail (and formality) of their specification [98]; the spectrum ranges from informal to formal ontologies, where: *Informal* ontologies include: controlled vocabularies (or catalogues), glossaries, thesauri and informal taxonomies; e.g. Wikipedia's[1] classification system, Wordnet [101]. *Formal* ontologies include: formal taxonomies with added semantics such as formal instances, frames (properties), value restrictions, general logical constraints, disjoints etc. Many Semantic web ontologies fall within this category, e.g. Friend of a Friend (FOAF[2]) ontology, which models "people-related terms".

Ontological knowledge is formally represented by using an appropriate *ontology language* (i.e. a knowledge representation formalism). A machine-processable ontology language needs to fulfil a set of requirements: (*i*) provide *well-defined syntax* therefore knowledge is machine readable; (*ii*) provide *formal semantics* which gives precise descriptions of the meaning of the statements in an ontology; (*iii*) have *sufficient expressive power* in order to adequately capture the domain of interest; (*iv*) support *automated and efficient reasoning* over the knowledge base. There are numerous knowledge representation formalisms, such as the following (non-exhaustive) list of examples:

- *First-order logic* (FOL, or predicate logic) is a generalisation of propositional logic that that uses quantified variables over (non-logical) objects (i.e. predicates) [149]. For example, the following sentence, $\forall x(Mother(x) \rightarrow Woman(x))$, states that every mother is a women. Given the expressive freedom FOL provides in the choice of predicates and the use of variables, reasoning in FOL is sound but incomplete. The Knowledge Interchange Format (KIF) is based on FOL. KIF is a formal language that was intended to support interchange of knowledge between computer systems, it was one of the first formats to model ontologies [56].

- *Frames* are data structures that focus on explicit and intuitive representation of knowledge [22], e.g. the FOAF ontology. OIL [49] is another example of a frame-based language, among with its successor DAML+OIL [78], which was later developed into the OWL language.

- *Description logics* are a family of formal knowledge representation languages with model-theoretic semantics [5]. Due to the fact that DLs are decidable subsets of FOL, it is possible to cary out sound and complete (basic) reasoning tasks. DLs provide the foundation of OWL, the *de facto* standard of the Semantic Web [99].

2.2 Description Logics

This section first describes the architecture of a DL knowledge base, then introduces the syntax and the semantics of DLs (Section 2.2.1). Section 2.2.2 presents the basic

[1] `https://en.wikipedia.org/wiki/Wikipedia:About`
[2] `http://www.foaf-project.org`

reasoning services permitted on DL knowledge bases[3]. Section 2.2.3 discusses two DL languages, which underpin named OWL profiles that are described in later sections. Section 2.2.4 and Section 2.2.4 introduces two complex reasoning services: *modularisation*, which can be used as a space reduction mechanism in definability and MDS computation, and, *justification-based explanations*, which facilitate definability validation and the heuristic-based definition axiom rewriting approach presented here.

2.2.1 DL Architecture, Syntax and Semantics

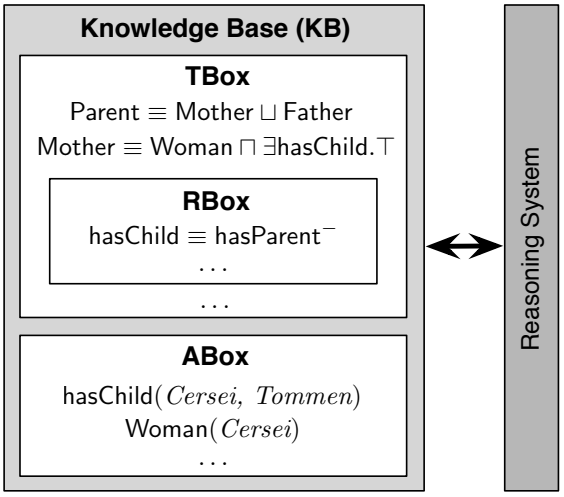


FIGURE 2.1: Description Logics based knowledge representation system architecture.

A DL knowledge base, depicted in Figure 2.1, consits of three main components:

- the *TBox* (terminological box, or schema) commonly denoted as $\mathcal{T}$, introduces the *terminology*, which defines the vocabulary of the conceptualised domain, by specifying how concepts and roles relate to each other;
- the *RBox* (role box) denoted as $\mathcal{R}$, is a subset of a TBox, that allows the representation of further, role-centric modelling features;
- the *ABox* (assertional box or data) denoted as $\mathcal{A}$, contains *assertions* about named individuals in terms of the vocabulary, i.e. it represents concrete data.

The basic building blocks used in DLs to conceptualise a particular domain of interest are the following:

- *atomic concepts*, or concept names (also called classes, denoted as $\mathsf{N}_\mathcal{C}$), e.g. Mother, Democracy. Concepts correspond to unary predicates in FOL.
- *atomic roles*, or role names (also referred to as properties, or relationships; denoted as $\mathsf{N}_\mathcal{R}$). Roles correspond to binary predicates in FOL.

[3]The presented materials are based on the book [7].

- *individuals*, also called as constants, or instances; denoted as $\mathsf{N}_\mathcal{I}$. Individuals correspond to constants in FOL.

An *entity* e is either a named concept, role, or individual. In addition, DLs denote the set of all objects in a domain with the concept $\top$ (often called 'thing'), and its complement $\bot$ ('nothing').

Signature. A signature is a finite set of concept, role and individual names (i.e. a finite set of entities). The signature $\mathsf{Sig(C)}$ denotes the signature of a complex concept C, which is the set of concept, role and individuals names that appear in its the description, for example:

$$\mathsf{Sig(Woman} \sqcap \exists \mathsf{hasChild}.\top) = \{\mathsf{Woman}, \mathsf{hasChild}\} \tag{2.1}$$

The $\mathsf{Sig(C} \sqsubseteq \mathsf{D})$ denotes the signature of a concept inclusion axiom (or $\mathsf{Sig}(\alpha)$, where $\alpha : C \sqsubseteq D$), which is defined as:

$$\mathsf{Sig(C} \sqsubseteq \mathsf{D}) = \mathsf{Sig(C)} \cup \mathsf{Sig(D)} \tag{2.2}$$

The signature $\mathsf{Sig}(\mathcal{T})$ denotes all entities of a given TBox $\mathcal{T}$, which is the union of all the signatures of its axioms, or the union of all concept, role, and individual names of the ontology. For example, given the TBox $\mathcal{T} = \{\mathsf{A} \sqsubseteq \mathsf{B}, \top \sqsubseteq \exists \mathsf{r}.\mathsf{C}\}$, the signature would be $\mathsf{Sig}(\mathcal{T}) = \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{r}\}$[4]. The cardinality of a signature (i.e. number of entities contained in the set) is denoted $|\mathsf{Sig(C)}|$.

Semantics. The semantics of description logics is defined formally in terms of an *interpretation* function $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$. The *interpretation domain* $\Delta^\mathcal{I}$ is a non-empty set and the *interpretation function* $\cdot^\mathcal{I}$ maps: each concept name $\mathsf{A} \in \mathsf{N}_\mathcal{C}$ to a subset $\mathsf{A}^\mathcal{I}$ of $\Delta^\mathcal{I}$; each role name $\mathsf{r} \in \mathsf{N}_\mathcal{R}$ to a binary relation $\mathsf{r}^\mathcal{I}$ on $\Delta^\mathcal{I}$; and each individual name a to an element $\mathsf{a}^\mathcal{I} \in \Delta^\mathcal{I}$.

Axioms. A DL *axiom* is a variable free well-formed formula, which states relationships between concepts, roles and individuals of the application domain. Axioms are fundamental modelling primitives of DL ontologies, such that an ontology is formalised as a finite set of axioms. At the architectural level, an axiom can either be categorised as a TBox, RBox or ABox axiom (Figure 2.1). Table 2.1 presents the syntax and semantics of DL axioms. In this thesis, axioms are denoted by Greek letters (potentially augmented with subscripts).

Concept Axioms. We can distinguish between two type of TBox concept axioms, w.r.t. the syntactic form of an axiom:

[4]Please note that the signature never includes any logical operators, or the top ($\top$) and bottom ($\bot$) concepts.

KB PART	$\mathcal{DL}$ *Axiom*	SYNTAX	SEMANTICS
TBox	Concept inclusion (CI)	$\mathsf{C} \sqsubseteq \mathsf{D}$	$\mathsf{C}^{\mathcal{I}} \subseteq \mathsf{D}^{\mathcal{I}}$
	Concept equality (CE)	$\mathsf{C} \equiv \mathsf{D}$	$\mathsf{C}^{\mathcal{I}} = \mathsf{D}^{\mathcal{I}}$
RBox	Role inclusion	$\mathsf{r} \sqsubseteq \mathsf{s}$	$\mathsf{r}^{\mathcal{I}} \subseteq \mathsf{s}^{\mathcal{I}}$
	Role equality	$\mathsf{r} \equiv \mathsf{s}$	$\mathsf{r}^{\mathcal{I}} = \mathsf{s}^{\mathcal{I}}$
	Complex role inclusion	$\mathsf{p} \circ \mathsf{r} \sqsubseteq \mathsf{s}$	$\mathsf{p}^{\mathcal{I}} \circ \mathsf{r}^{\mathcal{I}} \subseteq \mathsf{s}^{\mathcal{I}}$
	Role disjointness	$\mathsf{p} \sqcap \mathsf{r} \sqsubseteq \bot$	$\mathsf{p}^{\mathcal{I}} \cap \mathsf{r}^{\mathcal{I}} = \emptyset$
ABox	Concept assertion	$\mathsf{C}(a)$	$a^{\mathcal{I}} \in \mathsf{C}^{\mathcal{I}}$
	Role assertion	$\mathsf{r}(a, b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in \mathsf{r}^{\mathcal{I}}$
	Individual equality	$a \approx b$	$a^{\mathcal{I}} = b^{\mathcal{I}}$
	Individual inequality	$a \not\approx b$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$

TABLE 2.1: Syntax and semantics of Description Logics axioms.

- a *concept inclusion* (CI) axiom uses the subsumption ($\sqsubseteq, \sqsupseteq$) constructors, and describes containment relation between different (potentially complex, i.e. non-atomic) concepts, which denote diverse set(s) of individuals.
- a *concept equivalence* (CE) axiom uses the equality ($\equiv$) concept constructor, and describes different (potentially complex) concepts denoting the same set of individuals. A concept equivalence $\mathsf{C} \equiv \mathsf{D}$ is an abbreviation for the two concept inclusions $\mathsf{C} \sqsubseteq \mathsf{D}$ and $\mathsf{D} \sqsubseteq \mathsf{C}$.

For example, the following axiom:

$$\mathsf{Mother} \equiv \mathsf{Woman} \sqcap \exists \mathsf{hasChild}.\top \tag{2.3}$$

is an equivalence that describes the concept of Mother as a woman who has at least one child. The concept Mother could alternatively be characterised using two inclusion axioms (axioms 2.4 and 2.5):

$$\mathsf{Mother} \sqsubseteq \mathsf{Woman} \tag{2.4}$$

$$\mathsf{Mother} \sqsubseteq \exists \mathsf{hasChild}.\top \tag{2.5}$$

These inclusions state that a mother is a woman, and that a mother has a child, respectively.

In terms of the meaning of axioms, we can classify them as *definitions*, or *general concept inclusions (GCIs)*. A definition either takes the form of an equality ($\mathsf{C} \equiv \mathsf{D}$), or an inclusion ($\mathsf{C} \sqsubseteq \mathsf{D}$), where the left-hand side (LHS) concept within the definition must always be atomic. A GCI is an inclusion (or an equivalence [129]) axiom, where both the LHS and RHS concept is potentially complex. There are two type of definitions: *primitive* and *non-primitive*, where the former is always formalised as a CI axiom, and the latter is formalised as a CE axiom. The essential distinction between these axioms is that a definition either only provides the *necessary* condition, or provides both the *necessary and sufficient* conditions for describing a particular concept.

Definition 2.1 (Primitive concept definition)**.** A *primitive concept definition* is of the form $\mathsf{A} \sqsubseteq \mathsf{C}$, where A is atomic, and C is a potentially complex concept.

Primitive concept definitions allow C to model the necessary conditions for being A, which can range from a vague statement (e.g. axiom 2.3) to a rich description (e.g. $\mathsf{Mother} \sqsubseteq \exists \mathsf{hasChild}.\top \sqcap \mathsf{Woman} \sqcap \mathsf{Human\_being}$), however, in order to precisely define a concept one needs to known the complete set of necessary conditions. Considering a small ontology consisting of axioms 2.4 and 2.5 (above), it can be inferred that every mother is a woman who has a child. However, it *cannot* be inferred that these two conditions unambiguously conceptualise the concept of Mother under this ontology, due to the fact that Description Logics make the *open-world assumption*: Although these axioms may currently be all that is known about Mother, it does not mean that these statements define the concept in its entirety. A (non-primitive) concept definition represents the exact meaning of a concept, under the given ontology. For example, axiom (2.3) incorporates both axioms (2.4, 2.5), but also states the sufficient conditions to define the concept of Mother.

Definition 2.2 (Concept definition)**.** A *(non-primitive) concept definition* is of the form $\mathsf{A} \equiv \mathsf{C}$, where A is a concept name, C is a potentially complex concept.

Intuitively, in a concept definition (2.2), C describes the necessary and sufficient conditions for an individual to be an A.

Definitions are used to introduce *symbolic names* for complex concepts. Unless the concept definition is cyclic, these symbolic names can be used as abbreviations in more complex descriptions. Assuming Father is defined analogously to Mother as a Man who has a child, then Parent may be defined as:

$$\mathsf{Parent} \equiv \mathsf{Father} \sqcup \mathsf{Mother} \tag{2.6}$$

which is a succinct, and more human-readable version of the semantically equivalent definition axiom:

$$\mathsf{Parent} \equiv (\mathsf{Man} \sqcap \exists \mathsf{hasChild}.\top) \sqcup (\mathsf{Woman} \sqcap \exists \mathsf{hasChild}.\top) \tag{2.7}$$

Synonyms are special cases of concept definitions, where in a concept equality axiom there is a single concept name on both sides. For example, the axiom (2.8) states that both concepts, Mother and Mom, convey the same meaning under a given ontology, i.e. they denote the same set of individuals, in every interpretation of the domain:

$$\mathsf{Mother} \equiv \mathsf{Mom} \tag{2.8}$$

Concept definitions are also categorised as either *cyclic*, or *acyclic* on the bases of whether they directly, or indirectly use the defined concept to describe itself. An example of a definition containing a direct cycle is:

$$\mathsf{Human\_being} \equiv \exists \mathsf{hasParent}.\mathsf{Human\_being} \tag{2.9}$$

where the defined concept Human_being directly uses itself on the right-hand side of the non-primitive definition axiom.

Role axioms. A role denotes a binary relation between instances of concepts (i.e. individuals). Each role is described in terms of its corresponding *domain* and *range* concept: For example, the non-DL statement parentOf(Parent, Human_being) specifies the concepts Parent and Human_being, as the domain and range of the role parentOf, respectively. In DLs, this statement is expressed by the next two axioms (2.10); where the first axiom describes the concept of parent as the set of all individuals who are a parent of someone; and the latter axiom denotes human beings as the set of all individuals who have a parent:

$$\{\exists \mathsf{parentOf}.\top \sqsubseteq \mathsf{Parent}, \top \sqsubseteq \forall \mathsf{parentOf}.\mathsf{Human\_being}\} \tag{2.10}$$

These axioms both impose restrictions on the use of the role, and constrain the meaning of the concepts, but does not precisely *define* the meaning of the role. Roles are modelled by *RBox axioms* in a terminology. Similarly to concepts, there are two type of role axioms: *equivalence* and *inclusion*. For example, the following axiom (2.11):

$$\mathsf{brotherOf} \equiv \mathsf{maleSiblingOf} \tag{2.11}$$

states that the role names brotherOf and maleSiblingOf are synonyms, and therefore interchangeable within an ontology. Axiom (2.12) presents a role hierarchy, that captures that a parent of somebody is also its ancestor:

$$\mathsf{parentOf} \sqsubseteq \mathsf{ancestorOf} \tag{2.12}$$

The *characteristics of roles*, such as reflexivity, symmetry and transitivity are also expressed by RBox axioms, using either the equivalence, or the inclusion operator. A symmetric role (marriedTo) is equivalent to its own inverse, whereas an asymmetric role (parentOf) is disjoint from its inverse, i.e.:

$$\mathsf{marriedTo} \equiv \mathsf{marriedTo}^{-} \qquad \mathsf{parentOf} \sqcap \mathsf{parentOf}^{-} \sqsubseteq \bot \tag{2.13}$$

In the inclusion axioms (2.14), the role knows is reflexive, whereas marriedTo is an irreflexive relation:

$$\top \sqsubseteq \exists \mathsf{knows}.\mathsf{Self} \qquad \top \sqsubseteq \neg \exists \mathsf{marriedTo}.\mathsf{Self} \tag{2.14}$$

In addition to equivalence and inclusion axioms, role transitivity is described by the next non-DL axiom (2.15):

$$\mathit{transitive}(\mathsf{isPartOf}) \tag{2.15}$$

The following axiom (2.16) is an example of a *complex role* inclusion axiom (where $\circ$ denotes the role chain operator) that defines the role uncleOf as the composition of the

roles brotherOf and parentOf:

$$\mathsf{brotherOf} \circ \mathsf{parentOf} \sqsubseteq \mathsf{uncleOf} \tag{2.16}$$

Simple and Complex Entities. In addition to atomic entities (concepts and roles), *complex concepts* are built inductively using other concepts, roles, and individuals conjoined by concept constructors (logical operators), whereas *complex roles* are built inductively using roles and role constructors.

An atomic concept is an elementary description that either simply denotes a class of objects in the interpretation of the domain, and also provides an *abbreviation or name* for a complex description (i.e., a complex concept). For example, in an ontology, that consists of the single axiom $\mathsf{Mother} \equiv \mathsf{Woman} \sqcap \exists \mathsf{hasChild}.\top$, both Mother and Woman are atomic concepts, with the distinction that Mother assigns a name to the complex description on the right-hand side of the axiom, whereas Woman is simply a constituent entity of this description. A complex concept (or *compound* concept) denotes classes of objects in the interpretation of the domain. The complex concept $\mathsf{Woman} \sqcap \exists \mathsf{hasChild}.\top$ in the RHS of the CE axiom (2.3) provides the *meaning* of the Mother concept name in the LHS of this axiom.

A role is non-atomic (or non-simple [88]), if some complex role inclusion axiom (i.e. an inclusion whose left-hand side uses the role composition/role chain constructor $\circ$) implies instances of the role; otherwise it is atomic. For example, the axiom (2.16) describes the complex role uncleOf as the relation between instances of males (domain concept) and people (range concept) that have a sibling and that this sibling has a child (range concept). If the ontology contains no other axioms, other than domain and range restrictions, or corresponding role characteristic statements, then both brotherOf and parentOf are *simple* roles.

The semantics of non-atomic entities is then defined in terms of atomic concepts and roles. An interpretation $\mathcal{I}$ *satisfies* (or in other words *models*) an axiom α, denoted as $\mathcal{I} \models \alpha$ if:

$$\begin{aligned}
&\mathcal{I} \models \mathsf{C} \sqsubseteq \mathsf{D} \text{ iff } \mathsf{C}^{\mathcal{I}} \subseteq \mathsf{D}^{\mathcal{I}} \\
&\mathcal{I} \models \mathsf{C} \equiv \mathsf{D} \text{ iff } \mathsf{C}^{\mathcal{I}} = \mathsf{D}^{\mathcal{I}} \\
&\mathcal{I} \models \mathsf{r} \sqsubseteq \mathsf{s} \text{ iff } \mathsf{r}^{\mathcal{I}} \subseteq \mathsf{s}^{\mathcal{I}} \\
&\mathcal{I} \models \mathsf{r} \equiv \mathsf{s} \text{ iff } \mathsf{r}^{\mathcal{I}} = \mathsf{s}^{\mathcal{I}} \\
&\mathcal{I} \models \mathsf{C}(a) \text{ iff } a^{\mathcal{I}} \in \mathsf{C}^{\mathcal{I}} \\
&\mathcal{I} \models \mathsf{r}(a, b) \text{ iff } (a^{\mathcal{I}}, b^{\mathcal{I}}) \in \mathsf{r}^{\mathcal{I}}
\end{aligned} \tag{2.17}$$

Ontology. There are several ways to denote the notion of a DL ontology; for example, from an architectural perspective (Figure 2.1), the tuple $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ defines ontology $\mathcal{O}$ as its constituent TBox ($\mathcal{T}$) and ABox ($\mathcal{A}$). Another common way is given by the tuple, $\mathcal{O} = \langle \mathsf{Ax}(\mathcal{O}), \mathsf{Sig}(\mathcal{O}) \rangle$, which defines an ontology as the axioms it contains ($\mathsf{Ax}(\mathcal{O})$), and the set of entities that appear in its axioms ($\mathsf{Sig}(\mathcal{O})$).

Constructor	Syntax	Semantics
	Concepts	
Top concept	$\top$	$\Delta^{\mathcal{I}}$
Bottom concept	$\bot$	$\emptyset$
Atomic concept	A	$\mathsf{A}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
Atomic concept negation	$\neg\mathsf{A}$ $(\mathsf{A} \in \mathsf{N}_{\mathcal{C}})$	$\Delta^{\mathcal{I}} \setminus \mathsf{A}$
Complex concept negation	$\neg\mathsf{C}$	$\Delta^{\mathcal{I}} \setminus \mathsf{C}$
Concept intersection	$\mathsf{C} \sqcap \mathsf{D}$	$\mathsf{C}^{\mathcal{I}} \cap \mathsf{D}^{\mathcal{I}}$
Concept union	$\mathsf{C} \sqcup \mathsf{D}$	$\mathsf{C}^{\mathcal{I}} \cup \mathsf{D}^{\mathcal{I}}$
Limited existential restriction	$\exists\mathsf{r}.\top$	$\{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in \mathsf{r}\}$
Full existential restriction	$\exists\mathsf{r}.\mathsf{C}$	$\{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in \mathsf{r} \wedge b \in \mathsf{C}^{\mathcal{I}}\}$
Limited universal restriction	$\forall\mathsf{r}.\top$	
Full universal restriction	$\forall\mathsf{r}.\mathsf{C}$	$\{a \in \Delta^{\mathcal{I}} \mid \forall b.(a,b) \in \mathsf{r} \implies b \in \mathsf{C}^{\mathcal{I}}\}$
At most number rest. (unqualified)	$\leq_n \mathsf{r}$	$\{a \in \Delta^{\mathcal{I}} \mid \#\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in \mathsf{r}^{\mathcal{I}}\} \leq n\}$
At least number rest. (unqualified)	$\geq_n \mathsf{r}$	$\{a \in \Delta^{\mathcal{I}} \mid \#\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in \mathsf{r}^{\mathcal{I}}\} \geq n\}$
At most number rest. (qualified)	$\leq_n \mathsf{r}.\mathsf{C}$	$\{a \in \Delta^{\mathcal{I}} \mid \#\{b \in \mathsf{C}^{\mathcal{I}} \mid (a,b) \in \mathsf{r}^{\mathcal{I}}\} \geq n\}$
At least number rest. (qualified)	$\geq_n \mathsf{r}.\mathsf{C}$	$\{a \in \Delta^{\mathcal{I}} \mid \#\{b \in \mathsf{C}^{\mathcal{I}} \mid (a,b) \in \mathsf{r}^{\mathcal{I}}\} \geq n\}$
Nominal	$\{a\}$	$\{a^{\mathcal{I}}\}$
Nominal: one-of	$\{a_1, \ldots, a_n\}$	
Nominal: has value	$\exists\mathsf{r}.\{a\}$	
Datatypes		
	Roles	
Atomic role	r	$\mathsf{r}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
Inverse roles	$\mathsf{r} \equiv \mathsf{s}^{-}$	$\{(y,x) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (x,y) \in \mathsf{r}^{\mathcal{I}}\}$
Role hierarchy	$\mathsf{r} \sqsubseteq \mathsf{s}$	$\mathsf{r}^{\mathcal{I}} \subseteq \mathsf{s}^{\mathcal{I}}$
Transitive roles	$\mathsf{r}^{+} \sqsubseteq \mathsf{r}$	$\{x,y,z \in \Delta^{\mathcal{I}} \mid (x,y) \in \mathsf{r}^{\mathcal{I}}\}$

Table 2.2: Syntax and semantics of common DL concept and role constructors.

Constructors. A particular DL language is characterised by the set of *logical operators* for building complex concepts and roles, and the set of axioms for asserting statements about concepts, roles and individuals. Some more expressive constructors are associated with symbols, these are used for naming DL languages. For example in the $\mathcal{ALC}$ language (Attribute Language with Complements), the last letter "$\mathcal{C}$" stands for admitting the negation (of complex concept) operator[5]. Table 2.2 and present syntax and semantics of DL concept, and role constructors; where n is a nonnegative integer.

Further Semantics. The semantics of concept and role constructors are considered in terms of the interpretation function $\cdot^{\mathcal{I}}$, as shown in Table 2.2. A concept inclusion axiom $\mathsf{C} \sqsubseteq \mathsf{D}$ is *entailed* by a TBox $\mathcal{T}$ if and only if every model of $\mathcal{T}$ is also a model of $\mathsf{C} \sqsubseteq \mathsf{D}$, written as $\mathcal{T} \models \mathsf{C} \sqsubseteq \mathsf{D}$. One can also state that, for every interpretation $\mathcal{I}$, if $\mathcal{I} \models \mathcal{T}$, then $\mathcal{I} \models \mathsf{C} \sqsubseteq \mathsf{D}$.

An interpretation $\mathcal{I}$ *satisfies a TBox* $\mathcal{T}$ ($\mathcal{I} \models \mathcal{T}$) if and only if $\mathcal{I}$ satisfies every axiom $\alpha \in \mathcal{T}$. Similarly, $\mathcal{I}$ *satisfies an ABox* $\mathcal{A}$ ($\mathcal{I} \models \mathcal{A}$) iff every axiom $\alpha \in \mathcal{A}$ is true

[5]Note that some DLs do not follow the standard naming scheme, for instance $\mathcal{SHIF}(\mathcal{D})$ is commonly called as DL-Lite.

in $\mathcal{I}$. Furthermore, an interpretation $\mathcal{I}$ *is a model of the ontology* $\mathcal{O}$ ($\mathcal{I} \models \mathcal{O}$), where $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, if and only if it satisfies both $\mathcal{T}$ and $\mathcal{A}$. The satisfiability of concepts is given by the following definition:

Definition 2.3 (Satisfiable / Unsatisfiable Concept)**.** A concept C is *satisfiable* with respect to $\mathcal{T}$ if and only if there exists a model $\mathcal{I}$ of $\mathcal{T}$, where $\mathsf{C}^{\mathcal{I}} \neq \emptyset$ (i.e. $\mathcal{T} \not\models \mathsf{C} \sqsubseteq \bot$). A concept C is *unsatisfiable* with respect to $\mathcal{T}$, if there is a contradiction in $\mathcal{T}$ that implies that the concept cannot have any instances; i.e. for every model $\mathcal{I}$ of $\mathcal{T}$ it holds that $\mathsf{C}^{\mathcal{I}} = \emptyset$.

For example, given the following small ontology (2.18), the concept C is unsatisfiable, as it is the subclass of *disjoint* concepts A and B:

$$\mathcal{O} = \{\mathsf{A} \sqsubseteq \neg\mathsf{B}, \mathsf{C} \sqsubseteq \mathsf{A}, \mathsf{C} \sqsubseteq \mathsf{B}\} \tag{2.18}$$

There are two notable *errors* that an ontology can exhibit: *inconsistency*, and *incoherence*. These are defined as follows:

Definition 2.4 (Coherent / Incoherent Ontology)**.** An ontology $\mathcal{O}$ is *incoherent* if it contains at least one unsatisfiable concept, otherwise the ontology is called *coherent*.

Definition 2.5 (Consistent / Inconsistent Ontology)**.** An ontology $\mathcal{O}$ is *consistent* if and only if there exists a model $\mathcal{I}$ of $\mathcal{O}$. Conversely, $\mathcal{O}$ is *inconsistent* if there is no model $\mathcal{I}$ of $\mathcal{O}$.

In an inconsistent ontology, every axiom is entailed, i.e. $\mathcal{O} \models \top \sqsubseteq \bot$. Inconsistency is a severe error as it prevents the inference of meaningful knowledge from the ontology, thus rendering it unusable. An incoherent ontology can be consistent, and such ontologies can be (and often are) published, and used in applications. For example, the small ontology (2.18) is incoherent as it contains the unsatisfiable concept C, however it is consistent (it may have a model). This ontology may become inconsistent if there is some model of $\mathcal{O}$ that contains an individual $\mathsf{C}(foo)$, which is classified under the concept C, hence foo is the member of both A and B, whose intersection is an empty set.

2.2.2 Reasoning Services

A DL-based knowledge representation system has the ability to perform sound and complete reasoning; i.e. it can infer implicit knowledge from the facts stated explicitly in the ontology. For example, consider an ontology $\mathcal{O} = \{\mathsf{A} \sqsubseteq \mathsf{B}, \mathsf{B} \sqsubseteq \mathsf{C}\}$: although the axiom $\mathsf{A} \sqsubseteq \mathsf{C}$ is not explicitly specified, it can be made explicit through reasoning. The capability of inferring additional knowledge increases the modelling power of DL languages, and distinguishes them from weaker knowledge representation systems such as UML (Unified Modelling Language) [115].

As previously mentioned, DLs are decidable subsets of first-order logics. Unlike FOL, basic reasoning tasks are *decidable* for various less expressive DLs, meaning that there

exist algorithms which terminate and give the right answer (a boolean value, i.e. either a "yes", or a "no") after a finite number of steps. If an algorithm terminates within a reasonable amount of time (polynomial time), it can be regarded as *tractable*. For example, for the lightweight DL language $\mathcal{EL}$, deciding subsumption ($\mathcal{T} \models \mathsf{C} \sqsubseteq \mathsf{D}$) takes quadratic time, and thus is a tractable problem [5]. Whereas for more expressive DLs, such as $\mathcal{SHOIQ}$, even the basic inference services are intractable [5].

The purpose of an ontology is twofold: it should capture a given domain of interest as adequately (expressively) as possible; and it needs to be able to support inferences over the represented knowledge within a reasonable time-frame. However, the use of more expressive constructors increases the complexity of reasoning, with some DLs being intractable. Therefore, knowledge engineers can select from a range of different DL languages for their ontologies, but must find the best trade off between the level of expressivity required to model some domain, and the corresponding reasoning complexity (whilst still retaining tractability).

Reasoning services can be divided into two groups, terminological and assertional, on the basis of the architecture shown in Figure 2.1 (i.e. the TBox and ABox, respectively). Many standard inference problems can be reduced to testing (un)satisfiability of concepts [7]. Some reasoning services, such as the equivalence check, are simply a combination of other services. In the following, a list of standard reasoning services are described, grouped according to the KB "box" where the reasoning takes place:

Terminological (TBox) reasoning

- *Satisfiability.* Checking satisfiability of a concept determines whether a concept makes sense with respect to the knowledge base, i.e. if there exists an interpretation $\mathcal{I}$ of $\mathcal{T}$ where $\mathsf{C}^{\mathcal{I}}$ is nonempty.

- *Subsumption.* Checking whether a concept C is subsumed by (i.e. *contained in*) concept D with respect to $\mathcal{T}$ determines whether for every model $\mathcal{I}$ of $\mathcal{T}$, each instance of C is also an instance of D; $\mathcal{T} \models \mathsf{C} \sqsubseteq \mathsf{D}$ if $\mathsf{C}^{\mathcal{I}} \subseteq \mathsf{D}^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. This service is used for computing the concept hierarchy structure of an ontology by establishing sub- and super-concept relationships.

- *Equivalence.* Testing equivalence of two concepts C and D with respect to $\mathcal{T}$ shows whether they denote the exact same instances under every interpretation $\mathcal{I}$ of $\mathcal{T}$; i.e. $\mathcal{T} \models \mathsf{C} \equiv \mathsf{D}$ if $\mathsf{C}^{\mathcal{I}} = \mathsf{D}^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. It is used to determine whether two distinctly named concept, for instance Mother and Mom, represent the same meaning. This service is equivalent to a pairwise subsumption test, i.e. if $\mathcal{T} \models \{\mathsf{C} \sqsubseteq \mathsf{D}, \mathsf{D} \sqsubseteq \mathsf{C}\}$ then $\mathcal{T} \models \mathsf{C} \equiv \mathsf{D}$ also holds.

- *Disjointness.* Two concepts C and D are disjoint with respect to $\mathcal{T}$ if they share no instances under every interpretation $\mathcal{I}$ of $\mathcal{T}$; i.e. $\mathcal{T} \models \mathsf{C} \sqsubseteq \neg\mathsf{D}$ if $\mathsf{C}^{\mathcal{I}} \cap \mathsf{D}^{\mathcal{I}} = \emptyset$ for every model $\mathcal{I}$ of $\mathcal{T}$.

Assertional (ABox) reasoning

- *Consistency.* This service determines whether an ABox $\mathcal{A}$ is satisfiable with respect to some TBox $\mathcal{T}$. $\mathcal{A}$ is consistent if and only if there exists a model $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$.
- *Instance checking.* Given an individual a, a concept C, a TBox $\mathcal{T}$ and an Abox $\mathcal{A}$, this service determines whether a is an instance of C with respect to $\mathcal{T}$ for every model $\mathcal{I}$ of $\mathcal{T}$; i.e. $\mathcal{A} \models \mathsf{C}(a)$ iff $a^{\mathcal{I}} \in \mathsf{C}^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$.
- *Retrieval.* Given an ABox $\mathcal{A}$ and a concept C, this service queries the knowledge base to find all individuals a such that $\mathcal{A} \models \mathsf{C}(a)$.
- *Realisation.* Given an individual a and a set of concepts, this service finds the most specific concepts (the minimal concept with respect to the subsumption hierarchy) C from the set such that $\mathcal{A} \models \mathsf{C}(a)$.

Most modern (past 1990s) algorithms that perform DL reasoning services are based on the *tableau calculus* [5]. A tableau-based algorithm recursively and exhaustively applies *tableau rules* (competition rules) in an arbitrary order. This produces a finite representation of a model, i.e. a *constraint system*, which is a set of possible interpretations (ABoxes) of the model. The algorithm terminates when no further competition rule can be applied, or when all derived interpretations contain a clash.

A **reasoner** (reasoning engine, also commonly referred to as an external oracle) is a piece of software that implements automated inference services. For example, *HermiT* [63], *Pellet* [126], and *FaCT++* [140] are common, tableau-based DL reasoners. Most DL-based reasoners implement the previously listed standard reasoning services, some may even provide *non-standard* ones as well, such as computing the least common subsumer (LCS), computing justifications, modularisation etc.

In addition to simple instance retrieval, a DL knowledge base can be queried by using *conjunctive queries (CQs)* that permit more expressive queries than instance retrieval [62]. A DL conjunctive query is constructed by using logical operators: existential quantification ($\exists$), conjunction ($\wedge$); concept names and role names; and variable or individual names $(x_1, ..., x_k)$ that are either free or bound to a concept or role name (e.g. in the term $\mathsf{Student}(x)$, x is a variable bound to the concept name $\mathsf{Student}$). The following CQ retrieves tuples that show students, their primary supervisor and the major subject studied by the student:

$$(x, y, z) \leftarrow \mathsf{Student}(x) \wedge \mathsf{supervisorOf}(y, x) \wedge \mathsf{teaches}(y, z) \wedge \mathsf{studies}(x, z) \tag{2.19}$$

2.2.3 DL Languages

This thesis does not focus on any particular DL language (as previously mentioned, this is due to the fact that that implicit definability check works on DLs that do not accept Beth definability, hence MDSs are computable for such DLs). However, in order to better understand the syntax and semantics of a concrete language, now let us consider

two prototypical DL languages, the *lightweight* $\mathcal{EL}^{++}$ and the *very expressive* $\mathcal{SHOIN}$. These two languages also underpin two named OWL profiles (presented in Section 2.3); where A denotes a concept name, r a role name, a an individual name, and C, D denote possibly complex concepts in a given language:

- $\mathcal{EL}^{++}$ is a lightweight DL [6] designed to represent large-scale ontologies (e.g. biomedical ontologies such as SNOMED [15]) where the need for expressive axioms is limited, and that important inference problems such as the subsumption problem are decidable and tractable. $\mathcal{EL}^{++}$ supports the use of: concept intersection ($\sqcap$); full existential restriction ($\exists$); nominals ($\{a\}$); bottom concept ($\bot$); subsumption axioms; and concept equivalence axioms. Concepts in $\mathcal{EL}^{++}$ are formed according to the grammar:

  $$\mathsf{C} ::= \top \mid \ \bot \ \mid \mathsf{A} \mid \{\mathsf{a}\} \mid \mathsf{C} \sqcap \mathsf{D} \mid \exists \mathsf{r}.\mathsf{C}$$

  An $\mathcal{EL}^{++}$ TBox is a finite set of general concept inclusions of the form $\mathsf{C} \sqsubseteq \mathsf{D}$, and role inclusions of the form $\mathsf{r_1} \circ \ldots \mathsf{r_n} \sqsubseteq \mathsf{r}$, where $\mathsf{r}, \mathsf{r_1}, \ldots \mathsf{r}$ are role names, and n is a positive integer.

- $\mathcal{SHOIN}$ is a very expressive DL sublanguage that supports the use of: concept intersection ($\sqcap$); full existential restriction ($\exists$); nominals ($\{a\}$); bottom concept ($\bot$); number restrictions ($\leq_n \ \mathsf{r}, \geq_n \ \mathsf{r}$); transitive roles; and inverse roles (r^-). $\mathcal{SHOIN}$ concepts are formed according to the grammar:

  $$\mathsf{C} ::= \top \mid \ \bot \ \mid \mathsf{A} \mid \mathsf{C} \sqcap \mathsf{D} \mid \neg \mathsf{A} \mid \neg \mathsf{C} \mid \exists \mathsf{r}.\mathsf{C} \mid \forall \mathsf{r}.\mathsf{C} \mid \ \leq_n \mathsf{r} \mid \mathsf{r}^-$$

  An $\mathcal{SHOIN}$ TBox is a finite set of GCIs of the form $\mathsf{C} \sqsubseteq \mathsf{D}$, and role inclusions of the form $\mathsf{r} \sqsubseteq \mathsf{s}$.

2.2.4 Complex Reasoning Services

This section introduces two complex reasoning services that support various definability computation processes (presented in later chapters): *modularization* and *justification-based explanations*.

Ontology Modularization

An ontology *module* $\mathcal{M}$ is the relevant part of an ontology $\mathcal{O}$ (i.e. $\mathcal{M} \subseteq \mathcal{O}$) that is said to cover all the knowledge that $\mathcal{O}$ has about the entities in $\mathsf{Sig}(\mathcal{M})$ i.e. given an axiom α, $\mathcal{M}$ contains the same information about $\mathsf{Sig}(\alpha)$ as $\mathcal{O}$ and hence behaves the same way as $\mathcal{O}$ in all applications using the symbols in $\mathsf{Sig}(\alpha)$[6]. The process of extracting modules is referred to as *modularization* (or module extraction).

Definition 2.6 (Module)**.** Let $\mathcal{T}$ be a TBox. A subset $\mathcal{M}$ of $\mathcal{T}$ is a module of $\mathcal{T}$ iff:

$$\mathcal{M} \models \mathsf{C} \sqsubseteq \mathsf{D} \ \Leftrightarrow \ \mathcal{T} \models \mathsf{C} \sqsubseteq \mathsf{D}$$

[6]For a more formal definition and for further details on modularization, the reader is referred to [26].

for all concept inclusions $\mathsf{C} \sqsubseteq \mathsf{D}$ with $\mathsf{Sig}(\mathsf{C} \sqsubseteq \mathsf{D}) \subseteq \mathsf{Sig}(\mathcal{M})$.

Thus, a module "functions independently" from the TBox in the sense that it implies the same concept inclusions for its own subject matter (i.e. signature) as the whole TBox. Moreover, the TBox does not interfere with the module, i.e. it does not affect the meaning that the module defines for its own terms.

A module can be compared to the original ontology (or to any other modules, and ontologies) in terms of *logical difference*, with respect to a particular signature [87].

Definition 2.7. (Logical difference) Let $\mathcal{T}_1$, $\mathcal{T}_2$ be two TBoxes, and $\mathcal{S}$ a signature. The logical difference between $\mathcal{T}_1$ and $\mathcal{T}_2$, with respect to $\mathcal{S}$ consists of a set of subsumptions $\mathsf{C} \sqsubseteq \mathsf{D}$ with $\mathsf{Sig}(\mathsf{C} \sqsubseteq \mathsf{D}) \subseteq \mathcal{S}$ which follow from $\mathcal{T}_1$ but not from $\mathcal{T}_2$, or vice versa:

$$\begin{aligned}\mathsf{Diff}_{\mathcal{S}}(\mathcal{T}_1, \mathcal{T}_2) = \{\mathsf{C} \sqsubseteq \mathsf{D} | \mathcal{T}_1 \models \mathsf{C} \sqsubseteq \mathsf{D}, \mathcal{T}_2 \not\models \mathsf{C} \sqsubseteq \mathsf{D}, \mathsf{Sig}(\mathsf{C} \sqsubseteq \mathsf{D}) \subseteq \mathcal{S}\} \cup \\ \{\mathsf{C} \sqsubseteq \mathsf{D} | \mathcal{T}_2 \models \mathsf{C} \sqsubseteq \mathsf{D}, \mathcal{T}_1 \not\models \mathsf{C} \sqsubseteq \mathsf{D}, \mathsf{Sig}(\mathsf{C} \sqsubseteq \mathsf{D}) \subseteq \mathcal{S}\}\end{aligned}$$

From a logical viewpoint, $\mathcal{T}_1$ and $\mathcal{T}_2$ say the same about $\mathcal{S}$ if, and only if there is no logical difference between them, i.e. $\mathsf{Diff}_{\mathcal{S}}(\mathcal{T}_1, \mathcal{T}_2) = \emptyset$.

A module that represents knowledge about a *particular vocabluary* $\mathcal{S}$ (seed signature) is called an $\mathcal{S}$-module.

Definition 2.8. ($\mathcal{S}$-module) Let $\mathcal{O}$ be an ontology, $\mathcal{M}$ a module of $\mathcal{O}$, and $\mathcal{S}$ a signature. $\mathcal{M}$ is an $\mathcal{S}$-module of $\mathcal{O}$ if there is no logical difference between $\mathcal{M}$ and $\mathcal{O}$ with respect to $\mathcal{S}$, i.e. $\mathsf{Diff}_{\mathcal{S}}(\mathcal{M}, \mathcal{O}) = \emptyset$.

In this thesis, the $\mathcal{S}$-module extraction function is denoted as

$$\textsc{Mod}(\mathcal{O}, \mathcal{S}) \rightarrow \{\mathcal{M} | \mathcal{M} \subseteq \mathcal{O}\}$$

Locality Based Modules. Whilst there are various notions of inseparability applied to modules, this thesis is only concerned with LBMs for which implementations exist that can be used to compute modules. In recent years, ontology modularisation has been studied in great depth [26, 33, 34, 37, 118, 130], inter alia. Based on the notion of what axioms of an ontology are considered " relevant"[7] with respect to a given signature, module extraction techniques can be grouped into two main categories: *syntactic*, and *semantic*.

Syntactic (or structural) based approaches, as the name suggests, focus on the syntax of the axioms in the ontology and on the induced concept hierarchy, ignoring the semantics of the language. *Semantic* (or logic) based methods are concerned with preserving entailments that hold in the ontology, for a given signature. The disadvantage of logic-based module extraction is the cost of reasoning, which is often inefficient (above polynomial-time, and for more expressive DLs it is often exponential) [118]. This has led to the development of approaches that compute modules that are not necessarily minimal, by using syntactic approximations via locality.

[7]The notion of relevance is formally defined in [118] (inseparability relations).

Sattler et al. have noted that *syntactic-locality-based modules* (syntactic LBM) are suitable to (most) module extraction scenarios, because they preserve entailments and are efficiently extractable [118]. Furthermore, Del Vescovo et al. [34] have noted that for syntactic LBMs polynomial extraction algorithms are known[8], and widely used [75]. There are three main type of syntactic LBMs:

- $\bot$-module (bottom): includes axioms that define relationships between terms in $\mathcal{S}$ and more *general* terms in $\mathcal{O}$;
- $\top$-module (top): includes axioms that define relationships between terms in $\mathcal{S}$ and more *specific* terms in $\mathcal{O}$;
- $\top$ $\bot$ $*$-module (star): includes axioms that define and preserve relationships between terms in $\mathcal{S}$ by iterating the nesting of $\top$-extraction into $\bot$-extraction and vice versa.

Star-syntactic-LBMs preserve necessary entailments, and are the smallest out of the three types of modules.

Justification-based Explanations

A *justification* $\mathcal{J}$ for an entailment in an ontology is a set of TBox axioms ($\mathcal{J} \subseteq \mathcal{O}$) that is sufficient for that entailment to hold [73]. A justification is *minimal* if the entailment in question does not follow from any proper subset of the justification (i.e. no axiom can be removed without compromising the entailment).

Definition 2.9. $\mathcal{J}$ is a justification for $\mathcal{O} \models \alpha$ if $\mathcal{J} \subseteq \mathcal{O}$, $\mathcal{J} \models \alpha$ and for all $\mathcal{J}' \subset \mathcal{J}$ it is the case that $\mathcal{J}' \not\models \alpha$, where α is an axiom, and $\mathcal{J}, \mathcal{J}'$ are sets of axioms.

For example, consider a small ontology $\mathcal{O} = \{\mathsf{A} \sqsubseteq \mathsf{B}, \mathsf{B} \sqsubseteq \mathsf{C}, \mathsf{D} \sqsubseteq \exists \mathsf{r}.\mathsf{C}\}$ and an axiom $\alpha : \mathsf{A} \sqsubseteq \mathsf{C}$. $\mathcal{O} \models \alpha$ holds because $\{\mathsf{A} \sqsubseteq \mathsf{B}, \mathsf{B} \sqsubseteq \mathsf{C}\} \subseteq \mathcal{O}$, i.e. the entailment is justified. Horridge et. al. introduced an efficient approach that computes either a single, or all justifications of an entailment [73].

2.3 Web Ontology Language (OWL)

The Web Ontology Language is a Description Logics based knowledge representation language for authoring ontologies [99]. OWL was originally designed to support the Semantic Web [11]. OWL is the World Wide Web Consortium[9] recommendation for web applications of ontologies, since February 2004.

OWL provides the same modelling primitives as DLs, with the following terminology: *classes* (concepts), *properties* (roles), and *instances* (individuals). In OWL, properties are divided into *object properties* and *datatype properties*: the former denotes a relation

[8] The OWL API provides one off the shelf implementation of an LBM extraction algorithm that is polynomial.

[9] `https://www.w3.org/`

Constructor	DL Syntax	Example
owl:Thing	$\top$	
owl:Nothing	$\bot$	
intersectionOf	$\mathsf{C_1} \sqcap \ldots \sqcap \mathsf{C_n}$	$\mathsf{Man} \sqcap \mathsf{Parent}$
unionOf	$\mathsf{C_1} \sqcup \ldots \sqcup \mathsf{C_n}$	$\mathsf{Father} \sqcup \mathsf{Mother}$
complementOf	$\neg\mathsf{C}$	$\neg\mathsf{Man}$
oneOf	$\{x_1\} \sqcap \ldots \sqcap \{x_n\}$	$\{Tim\} \sqcap \{Tom\}$
allValuesFrom	$\forall\mathsf{p}.\mathsf{C}$	$\forall\mathsf{hasChild}.\mathsf{Woman}$
someValuesFrom	$\exists\mathsf{p}.\mathsf{C}$	$\exists\mathsf{hasChild}.\mathsf{Man}$
maxCardinality	$\leq_n \mathsf{p}$	$\leq_1 \mathsf{hasChild}$
minCardinality	$\geq_n \mathsf{p}$	$\geq_1 \mathsf{hasChild}$

Table 2.3: DL syntax of OWL constructors.

AxiomType	DL Syntax	Example
subClassOf	$\mathsf{C_1} \sqsubseteq \mathsf{C_2}$	$\mathsf{Father} \sqsubseteq \mathsf{Man}$
equivalentClass	$\mathsf{C_1} \equiv \mathsf{C_2}$	$\mathsf{Father} \equiv \mathsf{Man} \sqcap \mathsf{Parent}$
disjointWith	$\mathsf{C_1} \sqsubseteq \neg\mathsf{C_2}$	$\mathsf{Man} \sqsubseteq \neg\mathsf{Woman}$
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	$\{Harry\ Houdini\} \equiv \{Erik\ Weisz\}$
differentFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	$\{Tim\} \sqsubseteq \neg\{Tom\}$
subPropertyOf	$\mathsf{p} \sqsubseteq \mathsf{q}$	$\mathsf{hasFather} \sqsubseteq \mathsf{hasParent}$
equivalentProperty	$\mathsf{p} \equiv \mathsf{q}$	$\mathsf{hasCost} \equiv \mathsf{hasPrice}$
inverseOf	$\mathsf{p} \equiv \mathsf{q}^-$	$\mathsf{hasChild} \equiv \mathsf{hasParent}^-$
transitiveProperty	$\mathsf{p}^+ \sqsubseteq \mathsf{q}$	$\mathsf{hasAncestor}^+ \sqsubseteq \mathsf{hasAncestor}$
functionalProperty	$\top \sqsubseteq \leq_1 \mathsf{p}$	$\top \sqsubseteq \leq_1 \mathsf{hasFather}$
inverseFunctionalProperty	$\top \sqsubseteq \leq_1 \mathsf{p}^-$	$\top \sqsubseteq \leq_1 \mathsf{biologicalMotherOf}^-$

Table 2.4: DL syntax of OWL axioms.

between two individuals, the later is interpreted as a relation between an individual and a data value. A datatype is a unary predicate with a built-in interpretation, for example the *xsd:integer* datatype is interpreted as the set of all integer values [103]. Table 2.3 shows how the abstract OWL syntax corresponds to DL constructors, and Table 2.4 presents one possible XML-based OWL syntax of a DL concept, a role, and individual (i.e. assertion) axioms. OWL provides several different syntactic representations, such as the human readable Manchester syntax [77] (which is also used in the popular ontology editor, *Protégé*[10]), RDF/Turtle, or the RDF/XML-style syntax. The example depicted by Figure 2.2 shows how the DL statement $\mathsf{Parent} \equiv \mathsf{Father} \sqcup \mathsf{Mother}$ is represented in the RDF/XML syntax.

The first version of the language, *OWL 1.0*, was released in 2004 [99]. It had a web-based syntax, based on XML, RDF, and RDF Schema (RDF-S); moreover it supported the use of namespaces and URIs. The OWL 1.0 family consists of three increasingly expressive variants:

- *OWL Full* is a very expressive language, which is fully upward-compatible with RDF, however, reasoning for this language is undecidable.
- *OWL DL* is a sublanguage of OWL Full. It is essentially the DL $\mathcal{SHOIN}(\mathcal{D})$, which often permits reasonably efficient reasoning support, but it is not tractable.

[10]Protégé is a software by the Stanford Center for Biomedical Research, for developing and maintaining OWL ontologies; `http://protege.stanford.edu`

```
<owl:Class rdf:about="Parent">
  <owl:equivalentClass>
   <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
     <owl:Class rdf:about="Father"/>
     <owl:Class rdf:about="Mother"/>
    </owl:unionOf>
   </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```

FIGURE 2.2: OWL 2 statement example in RDF/XML syntax.

- *OWL Lite* is a sublanguage of OWL DL, without nominals and with XML datatypes, which corresponds to the DL $\mathcal{SHIF}(\mathcal{D})$. The language supports representing (large) classification hierarchies, such as thesauri and other taxonomies. Reasoning for OWL Lite is decidable, but not tractable.

OWL 2.0, the second (and current) version of the language, was released in 2009. It is an extension and revision of the previous version, providing full backward compatibility [66]. OWL 2.0 provides three new sublanguages (or profiles) that target different application scenarios:

- *OWL 2 EL* is based on the DL $\mathcal{EL}$++ (presented in Section 2.2.3), where reasoning services are performed in polynomial time (in the size of the ontology); this is best suited for ontologies with large vocabularies, making it a popular choice for medical knowledge bases or taxonomies.

- *OWL 2 QL* (Query Language) is based on the DL-Lite family [19], typically used for applications where the main purpose is efficient query answering. Thus the complexity of reasoning is performed in LOGSPACE w.r.t. the size of the ABox (i.e. the data asserted in the ontology).

- *OWL 2 RL* is a rule language (hence the acronym RL) inspired by Description Logic Programs (DLP [67]). OWL 2 RL is tailored towards applications that require high expressive power. Constructors such as existential quantification to a class, union and disjoint union to class expressions are not allowed in OWL 2 RL; this permits RL to be implemented using rule-based systems (e.g. Prolog).

In order to select a suitable OWL profile, an ontology engineer needs to consider the expressivity required by the application in order to sufficiently capture the given domain, the size of the datasets, and whether to prioritise instance or terminological reasoning.

There are several tools available for working with OWL ontologies, such as the *OWL API* [74], or the and *Jena API* [97]. The *OWL API* is a high level, Java-based Application Programming Interface (API) for working with OWL ontologies [74]. Since

its 2003 release, it has undergone several revisions supporting advances in OWL (most notably, the release of OWL 2.0), in addition to other design improvements. The OWL API is widely used in a variety of tools and applications, including in the experiment framework(s) of this thesis [74].

Part II

Minimal Definition Signatures (MDSs)

Chapter 3

Definability and Minimal Definition Signatures

This chapter introduces the notion of definability, and characterises *minimal definition signatures* that provide the base for employing implicit definability in semantic interoperability. The remainder of this chapter is organised as follows: Section 3.1 presents the notion of *concept and role definability*; Section 3.2 introduces definitions signatures and the notion of minimality in definition signatures; Section 3.3 provides *optimised algorithms for determining the definability status* of concepts and roles, and *validating definability*; Section 3.4 presents a *non-exhaustive list of definition patterns* that aim to generalise the frequent forms of creating definitions and aid in comprehension of definability (for the set of ontologies that were used in the empirical evaluation), moreover serve as input for a heuristic-based definition axiom generation approach. Lastly, Section 3.5 describes how MDSs can help in identifying certain ontology modelling errors.

3.1 Defining Concepts and Roles

As previously discussed in Section 2.2.1, concept definitions may contain cycles; these type of definitions are indeed valid and used in various scenarios. For example, the definition axiom $\mathsf{Human} \equiv \exists \mathsf{hasParent}.\mathsf{Human}$ defines a Human as somebody who has a parent who is also a Human. However, this work only concentrates on those cases where the concept name is defined without using the name itself (i.e. has a definition without a direct cycle), therefore *from now on we assume that in all definitions we do not allow the use of the defined concept name itself in the definition of the concept.* This restriction is motivated by the use case of this work, ontology alignment, whereby we want to replace the defined concept by a definition, or vice versa.

The *Beth definability theorem*, introduced for first-order logic by Beth in 1956, is a well-known property in classical logics that relates the notion of implicit definability to the one of explicit definability, by stating that a logical term is *implicitly definable* with respect to a theory if and only if it is also *explicitly definable* [12]. Given that explicit definability implies implicit definability, the Beth definability property holds for some

logic language $\mathcal{L}$ if the converse also holds, i.e. if implicit definability implies explicit definability. Consequently, if a term is implicitly defined then it is always possible to define it explicitly. As there are several variants of Beth definability [135], we focus on *Projective Beth definability* is a stronger formulation [72] with the ability to specify a set of terms, thus permitting us to restrict the vocabulary (i.e. signature) that can be used in definitions. Beth definability has also been studied in the context of DLs [135], where it has been used to compute explicit definitions based on implicit definitions. We thus assume a general DL language $\mathcal{L}$ for which the Beth definability property holds. This thesis builds on the definitions of concept definability, and the method for deciding implicit definability of concepts, presented by *ten Cate et al.* [135].

$$\begin{aligned}\mathcal{T}^{Family} = \{&\alpha_1 : \mathsf{Parent} \equiv \exists \mathsf{hasChild}.\top,\\ &\alpha_2 : \mathsf{Parent} \equiv \mathsf{Father} \sqcup \mathsf{Mother},\\ &\alpha_3 : \mathsf{Father} \sqsubseteq \mathsf{Man},\\ &\alpha_4 : \mathsf{Mother} \sqsubseteq \mathsf{Woman},\\ &\alpha_5 : \mathsf{Man} \sqsubseteq \neg\mathsf{Woman},\\ &\alpha_6 : \mathsf{hasChild} \equiv \mathsf{hasParent}^{-}\}\end{aligned} \tag{3.1}$$

The notion of definability, i.e. if a concept (or role) is considered to be 'defined' under an ontology, is not to be confused with whether a given concept (or role) is the subject of a primitive, or non-primitive definition axiom. A defined concept is assumed to be on the left-hand side, whilst its description is on the right-hand side of a non-primitive concept definition axiom[1] as illustrated by (3.2):

$$\underbrace{\mathsf{Parent}}_{\text{defined concept}} \equiv \underbrace{\mathsf{Mother} \sqcup \mathsf{Father}}_{\text{concept definition}} \tag{3.2}$$

A *defined concept* name is an abbreviation for a well-articulated definition; a concept is said to be associated with a precise meaning under a given ontology, if in any interpretation, the set of all individuals denoted by the concept can be unambiguously identified by either using only the *definition* of the concept, or just the concept name itself. However, this does not necessarily mean that a given defined concept is defined by a non-primitive concept definition axiom (e.g. $\mathsf{C} \equiv \mathsf{D}$) stated explicitly in the TBox, as concepts may be defined either *explicitly* or *implicitly*.

Explicit definability is a syntactic notion. Thus, defining a new concept C in an explicit way simply means describing it by a non-primitive definition or more precisely, a concept equivalence axiom whose left-hand side is the defined concept.

Beth definability has been introduced for arbitrary concepts, however in this thesis we *focus only on concept names* due to the motivating scenario, ontology alignment,

[1] This thesis assumes that all non-primitive concept definition axioms are presented in such way that the defined concept is always on the LHS, i.e. the axiom $\mathsf{A} \sqcup \mathsf{B} \equiv \mathsf{C}$ is normalised as $\mathsf{C} \equiv \mathsf{A} \sqcup \mathsf{B}$

which only requires concept names. Therefore we present a restricted version of ten Cate's definition [135] and define an explicitly defined concept name as:

Definition 3.1 (Explicitly defined concept [135])**.** Let C be a concept name, and $\mathcal{T}$ a TBox, and Σ a signature, where $\mathsf{C} \in \mathsf{Sig}(\mathcal{T})$, and $\Sigma \subseteq \mathsf{Sig}(\mathcal{T}) \setminus \{\mathsf{C}\}$. C is explicitly definable from Σ under $\mathcal{T}$, if and only if there is a potentially complex concept D such that $\mathcal{T} \models \mathsf{C} \equiv \mathsf{D}$ and $\mathsf{Sig}(\mathsf{D}) \subseteq \Sigma$.

For example, let us consider $\mathcal{T}^{Family}$ (3.1), a small $\mathcal{ALC}$-TBox describing the family domain. The concept Parent is the only explicitly defined concept in the ontology. Parent, defined by the axioms α_1 and α_2, has two semantically equivalent definitions:

$$\mathsf{Parent} \equiv \exists \mathsf{hasChild}.\top \equiv \mathsf{Father} \sqcup \mathsf{Mother} \tag{3.3}$$

In contrast to explicit definability, implicit definability is a semantic notion. Implicit definability means that although the definition of an implicitly defined concept is not asserted in the ontology, an explicit definition can be obtained for the given concept, or in other words *every implicitly defined concept is also explicitly definable*. A concept C being implicitly defined is a result of a set of general inclusion axioms that entail an explicit definition, e.g. $\{\mathsf{C} \sqsubseteq \mathsf{D}, \mathsf{D} \sqsubseteq \mathsf{C}\} \models \mathsf{C} \equiv \mathsf{D}$. We define implicitly definable concepts as:

Definition 3.2 (Implicitly definable concept [135])**.** Let C be a concept name, $\mathcal{T}$ a TBox, and Σ a signature, where $\mathsf{C} \in \mathsf{Sig}(\mathcal{T})$, and $\Sigma \subseteq \mathsf{Sig}(\mathcal{T})$. C is implicitly definable from Σ under $\mathcal{T}$ if and only if for any two models $\mathcal{I}$ and $\mathcal{J}$ of $\mathcal{T}$ such that

- $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$ and,
- for all entities $\mathsf{e} \in \Sigma$, $\mathsf{e}^{\mathcal{I}} = \mathsf{e}^{\mathcal{J}}$

then it holds that $\mathsf{C}^{\mathcal{I}} \equiv \mathsf{C}^{\mathcal{J}}$.

This definition relates definability to its model-theoretic semantics; it states that the interpretation of a particular concept C (i.e. the set of all individuals denoted by C in any model) under a TBox depends only on the extension of the entities in the specified signature (Σ) and the domain of discourse (i.e. Δ, the set of all individuals of in a given model). Given the example (3.1), it can be seen that both Mother and Father are implicitly defined concepts in $\mathcal{T}^{Family}$, as it is possible to express them by an explicit definition:

$$\mathsf{Father} \equiv \mathsf{Parent} \sqcap \neg\mathsf{Mother} \qquad \mathsf{Mother} \equiv \mathsf{Parent} \sqcap \neg\mathsf{Father} \tag{3.4}$$

Implicit definability can be challenging for human to comprehend; the definition axioms presented in 3.4 follow from $\mathcal{T}^{Family}$ because:

- $\alpha_2 \models \mathsf{Father} \sqsubseteq \mathsf{Parent}$ and $\alpha_2 \models \mathsf{Mother} \sqsubseteq \mathsf{Parent}$;
- $\alpha_3 : \mathsf{Father} \sqsubseteq \mathsf{Man}$, $\alpha_4 :\models \mathsf{Mother} \sqsubseteq \mathsf{Woman}$, and $\alpha_5 :\models \mathsf{Man} \sqsubseteq \neg\mathsf{Woman}$;

- $\{\alpha_3, \alpha_4, \alpha_5\} \models \mathsf{Father} \sqsubseteq \neg\mathsf{Mother}$, thus α_2 defines Parent as a disjoint union of Father and Mother.

An explicit definition is always intentional, i.e. it is formalised when something can be defined precisely; while implicit definitions are often unintentional. For instance, in $\mathcal{T}^{Family}$, Parent is explicitly defined by axiom α_2 as somebody who has a child, which is a more natural way to describe the concept than the following implicit definition,

$$\mathsf{Parent} \equiv \exists \mathsf{hasParent}^{-}.\top \tag{3.5}$$

which describes Parent as somebody who is a parent of someone. This example also demonstrates that some concepts may be defined both explicitly and implicitly.

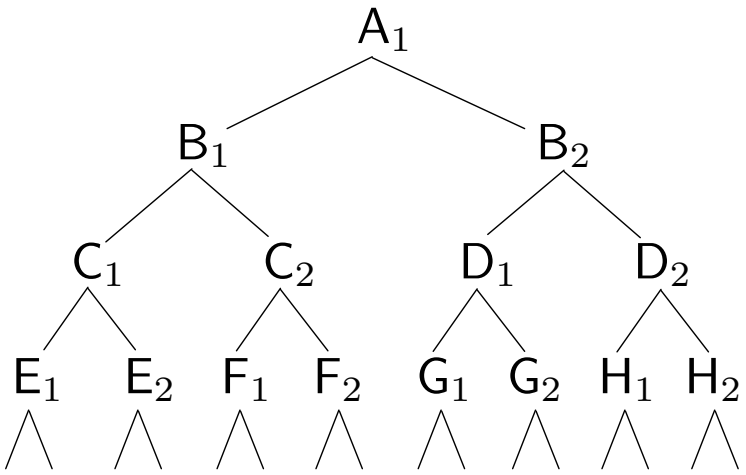


FIGURE 3.1: The number of definitions of a defined concept is exponential in the size of the ontology.

Quantifying definability. *The number of possible definitions* of a defined concept, regardless of whether it is explicitly or implicitly defined, is exponential in the size of the ontology. Thus the number of different definitions may serve as a *measure to quantify the extent of concept definability*. Defintions of defined concepts (i.e. the right-hand side of a non-primitive concept definition axiom) are built inductively using other, potentially defined concepts. By applying *unfolding* [20], each defined concept can be substituted with their definitions; as shown by Example (3.1). Therefore the number of possible concept defintions is dependent on the definability of its constituent concepts. As the definability of any defined concept is dependent on the definability of its own description, definability is therefore a recursive notion.

Example 3.1 (Number of definitions)**.** *Let us consider an example TBox $\mathcal{T}$, as illustrated in Figure 3.1, where all concepts are explicitly defined by the union of its constituent concepts.*

$$\begin{aligned}\mathcal{T} = \{\ & \mathsf{A}_1 \equiv \mathsf{B}_1 \sqcup \mathsf{B}_2, \\ & \mathsf{B}_1 \equiv \mathsf{C}_1 \sqcup \mathsf{C}_2, \\ & \mathsf{B}_2 \equiv \mathsf{D}_1 \sqcup \mathsf{D}_2, \\ & \mathsf{C}_1 \equiv \mathsf{E}_1 \sqcup \mathsf{E}_2, \\ & \mathsf{C}_2 \equiv \mathsf{F}_1 \sqcup \mathsf{F}_2, \\ & \mathsf{D}_1 \equiv \mathsf{G}_1 \sqcup \mathsf{G}_2, \\ & \mathsf{D}_2 \equiv \mathsf{H}_1 \sqcup \mathsf{H}_2, \\ & \ldots, \\ \}\end{aligned} \tag{3.6}$$

Step 1 *Concept A_1 is defined as*

$$\alpha_0 : \mathsf{A}_1 \equiv \mathsf{B}_1 \sqcup \mathsf{B}_2$$

where the definition of each describing concept (B_1 and B_2) can be further unfolded as:

$$\alpha_1 : \mathsf{A}_1 \equiv (\mathsf{C}_1 \sqcup \mathsf{C}_2) \sqcup (\mathsf{D}_1 \sqcup \mathsf{D}_2)$$

Only considering the unfolding of B_1 and B_2, we can see that A_1 can be rewritten into 4 different definitions as follows:

- $\mathsf{A}_1 \equiv \mathsf{B}_1 \sqcup \mathsf{B}_2$
- $\mathsf{A}_1 \equiv (\mathsf{C}_1 \sqcup \mathsf{C}_2) \sqcup \mathsf{B}_2$
- $\mathsf{A}_1 \equiv \mathsf{B}_1 \sqcup (\mathsf{D}_1 \sqcup \mathsf{D}_2)$
- $\mathsf{A}_1 \equiv (\mathsf{C}_1 \sqcup \mathsf{C}_2) \sqcup (\mathsf{D}_1 \sqcup \mathsf{D}_2)$

Step 2 *As all concepts on the right-hand-side of α_1 ($\mathsf{C}_1, \mathsf{C}_2, \mathsf{D}_1$ and D_2) are also defined, the definition α_1 can be further unfolded as:*

$$\alpha_2 : \mathsf{A}_1 \equiv ((\mathsf{E}_1 \sqcup \mathsf{E}_2) \sqcup (\mathsf{F}_1 \sqcup \mathsf{F}_2)) \sqcup ((\mathsf{G}_1 \sqcup \mathsf{G}_2) \sqcup (\mathsf{H}_1 \sqcup \mathsf{H}_2))$$

Assuming that all concepts on the right-hand-side of α_2 are also defined, unfolding can be repeated again, yielding more different definitions of A_1. Unfolding can be repeated until all concepts in a definition are undefined (atomic). Thus the number of possible definitions of A_1 is exponential in the size of $\mathcal{T}$.

3.1.1 Role definability

Although it has not been explicitly stated in the literature, the notion of definability extends to roles; hence a role can be classified either as *defined* (explicitly or implicitly), or as an *undefined* role. A defined role means that its extensions (set of pairs of individuals denoted by the role in an interpretation) can be unambiguously determined under an ontology, if the individuals of those entities that are used to define the role are known in an interpretation. Roles are described by specifying a domain and a range concept, however (in rewriting), concepts names are insufficient to precisely identify the individuals denoted by the defined role in an interpretation.

Similarly to concepts, explicit role definability is a syntactic notion, expressed by a *single axiom*; whereas implicit definability is a sematic notion where the meaning is implied by a *set of axioms*. In the axiom $\mathsf{r} \equiv \mathsf{s}$ both roles r and s are explicitly defined as synonyms. Moreover, the following two role inclusion axioms $\{\mathsf{r} \sqsubseteq \mathsf{s}, \mathsf{s} \sqsubseteq \mathsf{r}\}$ also define both roles as synonyms, however in this case these are considered to be implicitly defined entities. Another common role definition form is inverse, for instance, in $\mathcal{T}^{Family}$ (3.1) hasChild and hasParent are explicitly defined as inverse relations. The following axiom set explicitly defines the role parentOf:

$$\{\mathsf{parentOf} \equiv \mathsf{fatherOf} \sqcup \mathsf{motherOf},\ \ \mathsf{fatherOf} \sqcap \mathsf{motherOf} \sqsubseteq \bot\} \tag{3.7}$$

as well as implicitly defines both fatherOf and motherOf as

$$\mathsf{motherOf} \equiv \mathsf{parentOf} \sqcap \neg\mathsf{fatherOf} \qquad \mathsf{fatherOf} \equiv \mathsf{parentOf} \sqcap \neg\mathsf{motherOf} \tag{3.8}$$

The previous examples (3.7 and 3.8) shown roles being defined using exclusively using other role names implied by only RBox axioms. The following example presents a case where roles are implcitly definable under an ontology (comprising of TBox and ABox axioms) using nominals:

Example 3.2 (Roles defined under an ontology)**.** *The axioms in* $\mathcal{O}$ *entail that* $\mathsf{r} \equiv \mathsf{s}$*, i.e. the roles* r *and* s *are implicitly defined as synonyms.*

$$\begin{aligned} \mathcal{O} = \{\ & \top \equiv \exists \mathsf{s}.\{\mathsf{a}\}, \\ & \top \equiv \exists \mathsf{s}^{-}.\{\mathsf{b}\}, \\ & \top \equiv \exists \mathsf{r}.\{\mathsf{c}\}, \\ & \top \equiv \exists \mathsf{r}^{-}.\{\mathsf{d}\}, \\ & \{\mathsf{a}\} \equiv \{\mathsf{c}\}\} \end{aligned} \tag{3.9}$$

In practice, the most prevalent forms of defined roles are *synonym* and *inverse* roles. Role constructors, such as role hierarchy, transitive role, complex role inclusion axioms

are only part of (very) expressive DL languages, where the complexity of reasoning services considerably increases, making these languages less practical to use, and therefore less prevalent in real-world scenarios, compared to weaker DLs.

3.2 Minimal Definition Signatures (MDSs)

In this thesis, Σ *refers to a definition signature (DS)*, i.e. the set of entities that implicitly define a given concept (or role). A concept definition signature can be defined as:

Definition 3.3 (Concept Definition Signature (CDS))**.** A set of entities Σ is a *definition signature* of the *concept* C under a TBox $\mathcal{T}$, if and only if C is implicitly definable from Σ under $\mathcal{T}$ where $\mathsf{C} \in \mathsf{Sig}(\mathcal{T})$, and $\Sigma \subseteq \mathsf{Sig}(\mathcal{T}) \setminus \{\mathsf{C}\}$.

Similarly, a role definition signature can be defined as:

Definition 3.4 (Role Definition Signature (RDS))**.** A set of entities Σ is a *definition signature* of the *role* r under an TBox $\mathcal{T}$, if and only if r is implicitly definable from Σ under $\mathcal{T}$ where $\mathsf{r} \in \mathsf{Sig}(\mathcal{T})$, and $\Sigma \subseteq \mathsf{Sig}(\mathcal{T}) \setminus \{\mathsf{r}\}$.

As definition signatures may contain redundant members, their size could be very large. At the worst case, a DS is almost the size of the ontology signature, i.e. $\mathsf{Sig}(\mathcal{T}) \setminus \{\mathsf{C}\}$, as by definition the signature of the TBox is a DS of any constituent concept and role. Thus we introduced the notion of signature *minimality*:

Definition 3.5 (Minimal Definition Signature (MDS))**.** A signature Σ is a *minimal definition signature* of a defined entity e under a TBox $\mathcal{T}$, if there exists no other definition signature Σ' such that $\Sigma' \subset \Sigma$.

The minimality property of an MDS minimises the size of the signatures, by eliminating superfluous entities. However, a defined entity may have multiple unique MDSs under an ontology, with the same cardinality. The set of all MDSs of a given entity may overlap (where the difference of any two MDSs is not an empty set), or MDSs may be pairwise disjoint. From the definitions, it follows that every MDS is also a DS, and any DS may contain at least one, but potentially many MDSs.

Figure 3.2 presents the complete set of MDSs of all defined concepts and roles of the previously introduced $\mathcal{T}^{Family}$ example; moreover it shows the definition axioms and the corresponding justifications, which entail the definability. The signature $\Sigma = \{\mathsf{hasChild}, \mathsf{Man}, \mathsf{Woman}\}$ is a DS of all three defined concepts in the TBox (Parent, Mother, Father). However, this signature is not a minimal DS of Parent, because Parent can be defined by the following MDSs: $\{\mathsf{Father}, \mathsf{Mother}\}, \{\mathsf{hasChild}\}, \{\mathsf{hasParent}\}$; as entailed by the justifications $\mathcal{J}_4, \mathcal{J}_5$ and $\mathcal{J}_6$, respectively. It can be seen that a given defined entity may have many different MDSs, where the cardinality may differ significantly, depending on the definition type; e.g. Parent is explicitly defined by axiom α_2, whose MDS consists of two members, however it is also defined by two other MDSs that contain only a single entity. Furthermore, both Mother and Father have nine unique definition axioms, these depend

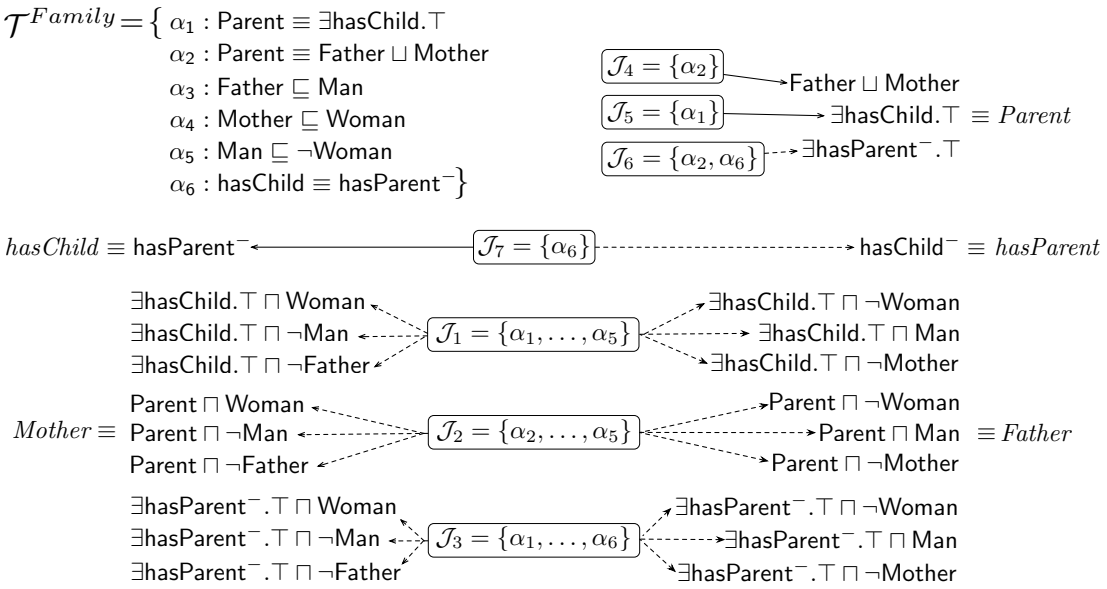


FIGURE 3.2: This small ontology describes a family domain. Concepts Mother and Father are only implicitly defined in $\mathcal{T}^{Family}$, hence these are also explicitly definable, while concept Parent is both explicitly and implicitly defined as shown by the definition axioms. Each definition axiom is explained by a justification ($\mathcal{J}_1 - \mathcal{J}_7$), where dashed line denotes implicit, normal line denotes explicit definability.

on the definability of the definition signature entities; for instance, removing axiom α_6 from the TBox (which provides the definition of the role hasParent and subsequently, an alternative definition of Parent) results in the loss of one MDSs for Parent and three MDSs for both Mother and Father.

Validity. A given signature Σ is a valid MDS if it exhibits *all* of the following properties:

- *correct*, i.e. Σ is a DS which explicitly or implicitly defines a concept (or role) under a given TBox;
- *minimal*, i.e. contains no redundant members;
- *not empty,* the only case when a concept (or role) can be defined with an empty signature is it is equivalent to $\top$ or $\bot$.

3.2.1 Quantifying MDSs

The number of possible MDSs of defined entities is potentially exponential in the size of the ontology. In order to justify this claim, the reader is referred back to Example 3.1 that shows that the number of different definitions of a defined concepts is potentially exponential in the size of the ontology. In this example, after applying unfolding at the first level (i.e. step 1) to the definition of concept A_1 can be rewritten into 4 different definitions. In addition, each of these definition have such a signature that is minimal, i.e. these are all different MDSs:

- $\mathsf{A}_1 \equiv \mathsf{B}_1 \sqcup \mathsf{B}_2$ $\quad \Sigma_1^{\mathsf{A}_1} = \{\mathsf{B}_1, \mathsf{B}_2\}$
- $\mathsf{A}_1 \equiv (\mathsf{C}_1 \sqcup \mathsf{C}_2) \sqcup \mathsf{B}_2$ $\quad \Sigma_2^{\mathsf{A}_1} = \{\mathsf{C}_1, \mathsf{C}_2, \mathsf{B}_2\}$
- $\mathsf{A}_1 \equiv \mathsf{B}_1 \sqcup (\mathsf{D}_1 \sqcup \mathsf{D}_2)$ $\quad \Sigma_3^{\mathsf{A}_1} = \{\mathsf{B}_1, \mathsf{D}_1, \mathsf{D}_2\}$
- $\mathsf{A}_1 \equiv (\mathsf{C}_1 \sqcup \mathsf{C}_2) \sqcup (\mathsf{D}_1 \sqcup \mathsf{D}_2)$ $\quad \Sigma_4^{\mathsf{A}_1} = \{\mathsf{C}_1, \mathsf{C}_2, \mathsf{D}_1, \mathsf{D}_2\}$

Further unfolding the defined concepts in definitions, the number of different definitions grows exponentially in the size of the ontology; and because each definition has a different signature, in this case the number of different MDSs is also exponentially in the size of the ontology signature.

It worth to note that the potential number of definitions and number of MDSs could be very different. For example both definitions $\mathsf{A} \equiv \mathsf{B}_1 \sqcup \mathsf{B}_2$ and $\mathsf{A} \equiv \mathsf{B}_2 \sqcup \mathsf{B}_1$ have the same MDS ($\Sigma^{\mathsf{A}} = \mathsf{B}_1, \mathsf{B}_2$), but they are syntactically different hence these are syntactically different explicit definitions.

3.3 Determining Definability

Deciding whether a concept has an explicit definition is a trivial task; whereas deciding whether an entity is implicitly defined is a potentially more complex process that requires reasoning, where the computational complexity is dictated by the DL language expressivity of the particular ontology (and subsequently the efficiency of the reasoning system). As part of this PhD research we aimed to provide pragmatic approaches for determining implicit and explicit definability. The remainder of this section presents the algorithms devised to check both types of definability as follows: First we address the process of determining explicit definitions, performed by Algorithm 1 presented in Section 3.3.1, followed by Algorithm 2 for deciding implicit definability (Section 3.3.2). Algorithm 3 provides an optimised way to decide the definability status of concepts (Section 3.3.3). Building on the implicit definability check (Algorithm 2), we devised an another approach (Algorithm 4) that identifies the implicit definability status of a given concept and also computes the corresponding *justification(s)*, thus explaining how a particular definability case is entailed in a TBox (Section 3.3.4). Role definability check is described in Section 3.3.5.

3.3.1 Determining Explicit Definitions

Algorithm 1 determines whether a particular concept name C is explicitly defined in a TBox $\mathcal{T}$, using a specified signature Σ. This process identifies both types of non-primitive concept definitions: (i) synonyms, where D is a concept name; and (ii) regular definitions, where D is complex concept. Please note that although this process is trivial, the algorithm is given for the purpose of referring to it in algorithms.

Furthermore, please note that this is different to determining explicit definability, as

- determining whether C has an explicit definition in $\mathcal{T}$ is a syntactic notion, detection is as simple as querying the TBox to find a non-primitive concept definition axiom of C, and checking whether the right-hand side signature is supported by Σ, i.e. whether $\mathsf{C} \equiv \mathsf{D} \in \mathcal{T}$;
- determining whether C is explicitly definable under $\mathcal{T}$ requires reasoning, i.e. whether $\mathcal{T} \models \mathsf{C} \equiv \mathsf{D}$

such that $\mathsf{Sig}(\mathsf{D}) \subseteq \Sigma$.

Algorithm 1: IsConceptExplicitlyDefined$(\mathsf{C}, \mathcal{T}, \Sigma)$

Input : C: concept name, $\mathcal{T}$: TBox, Σ: signature
Output: *Boolean*: True if C is explicitly defined by Σ in $\mathcal{T}$, given that $\mathsf{D} \in \mathcal{T}$ such that $\mathsf{Sig}(\mathsf{D}) \subseteq \Sigma$; and False otherwise.

```
1 if C ≡ D ∈ T, where Sig(D) ⊆ Σ then
2 |  return True
3 end
4 return False
```

3.3.2 Determining Implicit Definability

Algorithm 2 implements the method (presented in [134]) for determining implicit definability of a concept expressed in a given DL concept under an ontology, using a particular signature. This method reduces the implicit definability check to an entailment problem, and also provides bases for an alternative, more syntactic definition for implicit definability:

Theorem 3.6 (Implicit definability [134])**.** *Let* C *be a concept,* $\mathcal{T}$ *a TBox, and* Σ *a signature such that* $\Sigma \subseteq \mathsf{Sig}(\mathcal{T})$*, where* $\mathsf{C} \in \mathsf{Sig}(\mathcal{T})$*. Then* C *is implicitly definable from* Σ *under* $\mathcal{T}$ *if and only if* $\mathcal{T} \cup \mathcal{T}' \models \mathsf{C} \equiv \mathsf{C}'$*.*

Walkthrough.

1. First $\mathcal{T}'$, an identical copy of TBox $\mathcal{T}$ is created (line 1).
2. Next, the entity set $\mathcal{K}$ is initialised, by taking the signature of the TBox and reducing it with Σ, i.e. the set of entities allowed to be used in the complex concept which would define C (line 2).
3. At this point the process begins iterating through every occurrence of every entity in $\mathcal{T}'$ (i.e. in all axioms of the ontology) and renames those entities that also appear in $\mathcal{K}$, such that each entity $\mathsf{e} \in \mathcal{K}$ becomes e' in $\mathcal{T}'$. After each entity has been renamed $\mathsf{Sig}(\mathcal{T}) \cap \mathsf{Sig}(\mathcal{T}') = \Sigma$, i.e. the two TBoxes only share those entities that are allowed to be used in the definition of C.
4. Next the axiom $\mathsf{C} \equiv \mathsf{C}'$ is created, which is an equality between the two versions of the concept in question (i.e. the original in $\mathcal{T}$, and the renamed in $\mathcal{T}'$).

Algorithm 2: IsConceptImplicitlyDefined($\mathsf{C}, \mathcal{T}, \Sigma$)

Input : C: concept name, $\mathcal{T}$: TBox, Σ: signature
Output: *Boolean*: True if C is implicit defined, False otherwise

1 $\mathcal{T}' \leftarrow \mathcal{T}$
2 $\mathcal{K} \leftarrow \mathsf{Sig}(\mathcal{T}) \setminus \Sigma$
3 **for** $\mathsf{e} \in \mathcal{T}\prime$ **do**
4 **if** $\mathsf{e} \in \mathcal{K}$ **then**
5 $\mathsf{e} \leftarrow \mathsf{e}'$
6 **end**
7 **end**
8 **if** $\mathcal{T} \cup \mathcal{T}' \models \mathsf{C} \equiv \mathsf{C}'$ **then**
9 **return** True
10 **else**
11 **return** False
12 **end**

5. Finally the merged TBox $\mathcal{T} \cup \mathcal{T}'$ is queried to identify whether the axiom is entailed (line 8). If the entailment holds then C is indeed implicitly defined under $\mathcal{T}$, using only entities of Σ; otherwise it is undefined (by Σ).

Example 3.3. *(Algorithm 2 sample run)*
Let us consider a TBox $\mathcal{T}$ such that:

$$\mathcal{T} = \{\mathsf{C} \equiv \exists \mathsf{r}.\top, \mathsf{C} \equiv \mathsf{A} \sqcup \mathsf{B}, \mathsf{A} \sqsubseteq \mathsf{D}, \mathsf{B} \sqsubseteq \mathsf{E}, \mathsf{D} \sqsubseteq \neg \mathsf{E}\}$$

where the signature of $\mathcal{T}$ is

$$\mathsf{Sig}(\mathcal{T}) = \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}, \mathsf{r}\}$$

Given a concept $\mathsf{C} \in \mathcal{T}$, in order to determine whether $\Sigma = \{\mathsf{r}\}$ is a definition signature of C, Algorithm 2 performs the following steps:

1. *$\mathcal{T}'$, an identical copy of TBox $\mathcal{T}$ is created (i.e. $\mathcal{T}' = \mathcal{T}$)*

$$\mathcal{T}' = \{\mathsf{C} \equiv \exists \mathsf{r}.\top, \mathsf{C} \equiv \mathsf{A} \sqcup \mathsf{B}, \mathsf{A} \sqsubseteq \mathsf{D}, \mathsf{B} \sqsubseteq \mathsf{E}, \mathsf{D} \sqsubseteq \neg \mathsf{E}\}$$

2. *$\mathcal{K}$ is initialised as $\{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}, \mathsf{r}\} \setminus \{\mathsf{r}\} = \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}\}$*

3. *Each entity of $\mathcal{T}'$ such that $\mathsf{e} \in (\mathsf{Sig}(\mathcal{T})' \cap \mathcal{K})$ are renamed to e', hence*

   - $\mathcal{T}' = \{\mathsf{C}' \equiv \exists \mathsf{r}.\top, \mathsf{C}' \equiv \mathsf{A}' \sqcup \mathsf{B}', \mathsf{A}' \sqsubseteq \mathsf{D}', \mathsf{B}' \sqsubseteq \mathsf{E}', \mathsf{D}' \sqsubseteq \neg \mathsf{E}'\}$
   - $\mathsf{Sig}(\mathcal{T}) \cap \mathsf{Sig}(\mathcal{T}') = \{\mathsf{r}\}$

4. $\mathcal{T}' \cup \mathcal{T} = \{\mathsf{C} \equiv \exists \mathsf{r}.\top, \mathsf{C} \equiv \mathsf{A} \sqcup \mathsf{B}, \mathsf{A} \sqsubseteq \mathsf{D}, \mathsf{B} \sqsubseteq \mathsf{E}, \mathsf{D} \sqsubseteq \neg \mathsf{E},$
   $\mathsf{C}' \equiv \exists \mathsf{r}.\top, \mathsf{C}' \equiv \mathsf{A}' \sqcup \mathsf{B}', \mathsf{A}' \sqsubseteq \mathsf{D}', \mathsf{B}' \sqsubseteq \mathsf{E}', \mathsf{D}' \sqsubseteq \neg \mathsf{E}'\}$

5. $\mathcal{T}' \cup \mathcal{T} \models \mathsf{C} \equiv \mathsf{C}'$ *because*

- $\{\mathsf{C} \equiv \exists \mathsf{r}.\top, \mathsf{C}' \equiv \exists \mathsf{r}.\top\} \subseteq \mathcal{T}' \cup \mathcal{T}$
- $(\mathsf{C} \equiv \mathsf{C}') \quad \equiv \quad (\exists \mathsf{r}.\top)$

As the signature of the complex concept $\exists r.\top$ *is* Σ*, the algorithm determines that it is also a definition signature of* C*.*

Complexity. The computational complexity of the algorithm is mostly predicated on the complexity of the reasoning, i.e. the entailment check, which could range from polynomial to exponential, depending on the DL language expressivity of $\mathcal{T}$. As previously described, this is dedicated to an oracle and counts as a single step computation (i.e. one oracle call).

3.3.3 Determining Definability Type

Algorithm 3: IsConceptDefined($\mathsf{C}, \mathcal{T}$)

Input : C: concept name, $\mathcal{T}$: TBox
Output: type of definability, i.e. one of
{`Undefined, Explicitly, Implicitly`}

```
1 S ← Sig(T) \ {C}
2 if IsConceptExplicitlyDefined(C, T, S) then
3 |   return Explicitly
4 end
5 if IsConceptImplicitlyDefined(C, T, S) then
6 |   return Implicitly
7 end
8 return Undefined
```

This algorithm determines the *definability status* of a given concept under a TBox (defined either explicitly, implicitly, or undefined). For this purpose, we considered a signature that is the TBox signature, excluding the concept under consideration; i.e. the focus is on finding, whether there is a prospective definition signature of the concept, independently of the entities used in the definition signature, as all definition signature computation algorithms assume that the given concept must be definable.

In order to reduce complexity, the process is optimised by first checking explicit definability, due to the fact that potentially this approach has significantly lower computational complexity compared to the implicit definability check. The implicit definability check would also detect that a given concept is defined, although it would not provide a precise definability status. In case the concept in question is not explicit defined, then it is tested for implicit definability. If both definability checks fail, the concept is undefined in the ontology.

Algorithm 4: JustifyDefinability($\mathsf{C}, \mathcal{T}, \Sigma$)

Input : C: concept name, $\mathcal{T}$: TBox, Σ: signature
Output:

```
1  T' ← T
2  K ← Sig(T) \ Σ
3  for e ∈ T' do
4  |   if e ∈ K then
5  |   |   e ← e'
6  |   end
7  end
8  α ← C ≡ C'
9  J ←ComputeSingleJustification((T ∪ T'), α)
10 return J
```

3.3.4 Justifying Definability

Algorithm 2 is determines whether a given concept is implicit defined under an ontology, using a particular signature. However, due to its semantic nature, it is often difficult for humans, and especially in large ontologies, to comprehend how concepts are defined, and to identify the axioms used in a definition. *Justifications* [73] can be used to validate definability and to provide a set of axioms supporting an entailment[2]. The implicit definition check (Algorithm 2) works by testing whether a possible entailment holds. This makes it easy to extract the relevant axioms containing the implicit definition by extracting all justifications for that entailment, which is a standard service with off the shelf, highly optimized tooling [73].

In the scope of this research, the algorithm has two practical applications: (*i*) it aids validating the results of definition signature computation approaches by identifying a succinct axiom set; (*ii*) it provides an input for definition pattern recognition, which for a set of identified, non-exhaustive, patterns, makes possible spelling out the explicit definition of the concept (Section 3.4).

Walkthrough. In addition to deciding definability, this algorithm also identifies those *justification* axioms that correspond to the definitions of a defined concept. The algorithm follows the same procedure as its base version (Algorithm 2): first the input TBox is duplicated, then all the non-signature entities are renamed in the new TBox. At this point, instead of directly using a DL reasoner, the entailment check is delegated to the algorithm that computes a single justification (line 9), where $\mathcal{T} \cup \mathcal{T}'$ is the merged TBox, where the axiom α is tested for entailment[3]. If no justification is found (i.e. $\mathcal{J} = \emptyset$), the axiom is not entailed, meaning that the concept is not defined under $\mathcal{T}$ with the given signature.

[2] As described in Section 2.2.4, a justification is a (minimal) set of axioms that is sufficient for that entailment to hold.

[3] Justifications computation is aided by an external reasoner, i.e. an oracle.

Complexity. Compared to the base version, this algorithm has the added complexity of computing justification(s), which is a tractable process. However, when computing all justifications, it should be noted that in the worst case, the number of possible justifications for a given entailment could be exponential in the size of the ontology.

Optimisation. By default only a single justification is computed, however there is an option to compute all justifications of an entailment (using the ComputeAllJustifications algorithm, Chapter 4.1 in [73]). This is potentially useful for *heuristic-based axiom generation* (Section 3.4), which only supports a subset of possible definability cases, thus finding alternative justifications increases the chance that a supported case can be found.

3.3.5 Determining Role Definability

Determining explicit or implicit role definability can be achieved by using the same methods as for concepts (Algorithm 1 and 2). In order to decide definability of a concept whose description contains defined roles, there is no need to separately assess the definability of such roles. For example in $\mathcal{T}^{Family}$ (3.1), Parent is defined using the signature $\Sigma = \{\mathsf{hasParent}\}$, as it is formalised by the definition axiom (3.5), which is based on an inverse role definition. Thus the definability check for Parent with Σ is computed by the concept implicit definability algorithm.

Implicit definability check issue. During the empirical evaluation of MDS computation, it was noticed that when the TBox is used to decide role definability, false positives may emerge. For example, let us consider the small TBox, consiting of one axiom $\mathcal{T} = \{\exists \mathsf{s}.\top \equiv \exists \mathsf{r}.\top\}$, which implies that roles r and s share the same domain concept. In this case, roles r and s would be incorrectly identified as a definition axiom for both roles. This example was tested using several mainstream reasoners (*HermiT* [63] and *Pellet* [126]), however all provided the same false answer. At the time of writing it is unclear whether this is a bug in the implementation of the definability check algorithm or the reasoners[4], however in order to avoid this issue, implicit definability check for roles was implemented with the following modifications:

- The restricted *signature can only contain role names*, as roles are only defined in terms of other role names, i.e. $\Sigma \subseteq \mathsf{Sig}(\mathcal{R})$.
- Instead of operating on the entire TBox, the process is *restricted to the RBox* (set of role axioms).

Due to these restrictions the practical implementation would miss out on identifying cases of implicitly defined roles, where TBox and ABox axioms are used to define concepts, such as the one presented in the example 3.2.

[4] It is also possible that implicit definability check for roles requires a different algorithm than concepts, although for roles that are defined only by RBox axiom the algorithm provides correct results. Therefore this investigation is a suggested future work

Justifications. Analogously to concepts, in theory, role definability can be validated and explained by identifying the succinct set of axioms which entails definability. However, in practice this is currently unachievable due to the lack of existing tool support. Horridge et al. have released the *OWLExplanation API* [76] to facilitate computing entailments in Java applications, however, this does not support entailment for role axioms (e.g. $\mathcal{T} \models_? \mathsf{r} \equiv \mathsf{s}$).

3.4 Definition Patterns (DPs)

Entity type	Pattern ID	Pattern name
Concept	1	Implicit synonym concept
	2	Constituent concept of a disjoint union
	3	Domain concept of role
	4	Range concept of role
	5	Domain concept of synonym role
	6	Range concept of synonym role
	7	Domain concept of inverse role
	8	Range concept of inverse role
Role	9	Implicit synonym role
	10	Implicit inverse role

TABLE 3.1: List of concept and role definition patters

This section presents a non-exhaustive list of concept and role definition patterns. Although for the bigger picture of this thesis the exact definitions are not important (as previously discussed, we only focus on implicit definability and their MDSs as definitions themselves are not strictly required for the application of this research) it is still interesting to know what kind of explicit definitions can be found for implicitly definable concept and role names. *This is not a comprehensive study, i.e. the presented list is incomplete*, because we do not have an algorithm that would generate explicit definitions and we do not know for all ontologies used in the evaluation whether the Beth definability property stands, i.e. whether implicit definability implies explicit definability, which makes definition generation possible. Furthermore it was not the goal of this research to provide a comprehensive list of definition patterns.

As part of the empirical investigation of definability (presented in Chapter 5), we have computed the MDSs for numerous ontologies, and validated them by obtaining the corresponding justifications. The motivation of studying definition patterns was to find out what kind of implicit definitions we find that are not explicit. An explicit concept definition is always formalised as a single axiom, whereas the definition of an implicitly defined concept is derived from an axiom set (i.e. a justification); thus, implicit definitions are often not straightforward to recognise and interpret. *Although we do not have any rigorous methodology to derive such definitions, anecdotally some repeating some patterns were spotted during the empirical evaluation*: by studying the composition of MDSs (their cardinality, and the type and number of their member entities) together with their justifications (their size, and the type of their constituent axioms) we have

identified a number of *definition patterns (DPs)*. Some patterns are very obvious e.g. explicit definitions, and (#2) constituent concept of a disjoint union, however other patterns are much less intuitive e.g. (#1) and (#9) implicit concept and role synonyms. The list excludes explicit definitions.

These patterns can be used to analyse interesting ontologies, and to validate and interpret cases of implicit definability. For example, if a concept name is defined but it only has an implicit definition and no explicit definition, it can prompt the knowledge engineer to check whether the conceptualisation is correct. In addition the identifiable definition patterns permit a *heuristic-based axiom generation algorithms*, i.e. rewriting implicit definitions into explicit definitions, according to an inference rule, by processing a given MDS and a justification. For each pattern the following sections presents:

- *a matching pattern*, i.e. how we recognise that an MDS and its justification matches this pattern;
- *an extraction pattern*, i.e. how we form the definition (if there is one) from the MDS and the justification.

Heurisitc-based axiom generation algorithms requires the following steps:

1. testing concept or role definability;
2. computing an MDS if the given concept or role is defined;
3. obtaining a single justification for the MDS.

For the statistical evaluation of how often the identified patterns occur in the ontologies used for empirical evaluation please see Table 5.9 in Chapter 5, Section 5.2.2.

3.4.1 Concept Definition Patterns

***(#1)* Implicit synonym.**
Matching pattern

- Defined concept: C
- MDS: contains a single concept name ($|\Sigma| = 1$), which is the synonym of the defined concept, e.g. $\Sigma^{\mathsf{C}} = \{\mathsf{D}\}$
- Justification: contains more than one axiom ($|\mathcal{J}| \geq 2$), e.g. $\mathcal{J} = \{\mathsf{C} \sqsubseteq \mathsf{D}, \mathsf{D} \sqsubseteq \mathsf{C}\}$

Extraction pattern

- Definition: *Defined concept* is equivalent to the *single MDS member* concept name, e.g. $\mathsf{C} \equiv \mathsf{D}$

The simplest occurrence of an implicit synonym is when a concept is represented as two inclusion axioms:

$$\{\mathsf{C} \sqsubseteq \mathsf{D}, \mathsf{D} \sqsubseteq \mathsf{C}\} \models \mathsf{C} \equiv \mathsf{D}$$

As demonstrated by a real ontology example (3.4) implicit synonyms can be entailed by more complex justifications:

Example 3.4. [5] *In this ontology,* Plenary_lecture *is a synonym of* Tutorial.

$$\begin{aligned}\mathcal{J} = \{\ & \alpha_1 : \mathsf{Plenary\_lecture} \equiv \exists \mathsf{is\_given\_by}.\mathsf{Plenary\_lecture\_speaker},\\ & \alpha_2 : \mathsf{Plenary\_lecture\_speaker} \equiv \exists \mathsf{give}.\mathsf{Tutorial}\\ & \alpha_3 : \mathsf{Tutorial} \equiv \exists \mathsf{is\_given\_by}.\mathsf{Tutorial\_speaker}\\ & \alpha_4 : \mathsf{Tutorial\_speaker} \equiv \exists \mathsf{give}.\mathsf{Plenary\_lecture}\\ & \alpha_5 : \mathsf{give} \equiv \mathsf{is\_given\_by}^{-}\}\end{aligned}$$

In other cases, describing concepts as synonyms may be an unintended conceptual error, made by the ontology engineer, such as in the next example:

Example 3.5. [6] *Both concepts* Event *and* Document *are synonym terms of each other, however this may be an error, as in this case the concept names refer to semantically unrelated concepts.*

$$\begin{aligned}\mathcal{J} = \{\ & \alpha_1 : \mathsf{Event} \equiv \exists \mathsf{created\_by}.\mathsf{Person},\\ & \alpha_2 : \mathsf{Document} \equiv \exists \mathsf{created\_by}.\mathsf{Person}\}\end{aligned}$$

(#2) Constituent concept of a disjoint union.
A disjoint union states that a concept C is a union of a set of pairwise disjoint concepts $\{\mathsf{D}_1, \ldots, \mathsf{D}_n\}$. In this pattern, *all named and anonymous*[7] *concepts* are defined, where C is either a concept name, or a complex concept.
Matching pattern

- Defined concept: $\mathsf{D_j}$
- MDS:

$$\Sigma^{\mathsf{D}_j} = \{\ \underbrace{\mathsf{C},}_{\text{subsumer concept}}\ \underbrace{\mathsf{D}_1, \ldots \mathsf{D_i}, \mathsf{D_k}, \ldots, \mathsf{D_n}}_{\text{disjoint concepts}}\}$$

- Justification: consists of
  - a single concept equivalence axiom α_1, which states the union as the explicit definition of C;

[5] Conference corpus, *iasted.owl*
[6] Conference corpus, *paperdyne.owl*
[7] Assuming that any anonymous concept that appear in α_1 are not unfolded, for example if $\mathsf{C} \equiv \mathsf{D}_1 \sqcup \exists \mathsf{r}.\exists \mathsf{s}.\top$, then $\exists \mathsf{r}.\exists \mathsf{s}.\top$ is defined, but $\exists \mathsf{s}.\top$ is not.

- axioms $\alpha_2 - \alpha_m$ that state pairwise disjointness of the other signature entities;
- potentially, but not necessarily any other type of concept or role axioms;

e.g.:

$$\begin{aligned}\mathcal{J} = \{&\alpha_1 : \mathsf{C} \equiv \mathsf{D_1} \sqcup \ldots \sqcup \mathsf{D_n},\\ &\alpha_2 : \mathsf{D_i} \sqsubseteq \neg \mathsf{D_1},\\ &\ldots\\ &\alpha_m : \mathsf{D_i} \sqsubseteq \neg \mathsf{D_n},\\ &\ldots\}\end{aligned}$$

Extraction pattern

- Definition: to define any named concept D_j (of axiom α_1 RHS concepts) the required signature is $\Sigma^{\mathsf{D}_j} = \mathsf{Sig}(\alpha_1) \setminus \{\mathsf{D_j}\}$, where the definition axiom takes the form of D_j being defined as the conjunction of D, and the complement of the union of those concepts that are disjoint with D_j. $\mathsf{D_j} \equiv \mathsf{C} \sqcap \neg(\mathsf{D_1} \sqcup \ldots \sqcup \mathsf{D_i} \sqcup \mathsf{D_k} \sqcup \ldots \sqcup \mathsf{D_n})$;

where $(1 \leq i < j < k \leq n)$.

Example 3.6 illustrates this pattern:

Example 3.6. [8] $\mathsf{Regular\_contribution}$ *is explicitly defined by* α_1*, where the definition axiom signature is* $\Sigma^{\mathsf{Regular\_contribution}} = \{\mathsf{Extended\_abstract}, \mathsf{Paper}\}$*. On the RHS of* α_1 *there are two concept names, these are declared to be pairwise disjoint in axiom* α_2*.*

$$\begin{aligned}\mathcal{J} = \{&\alpha_1 : \mathsf{Regular\_contribution} \equiv \mathsf{Extended\_abstract} \sqcup \mathsf{Paper},\\ &\alpha_2 : \mathsf{Extended\_abstract} \sqsubseteq \neg\mathsf{Paper}\}\end{aligned}$$

Thus $\mathsf{Extended\_abstract}$ and Paper are implicitly defined from $\mathcal{J}$ as follows:

$$\begin{aligned}\mathsf{Extended\_abstract} &\equiv \mathsf{Regular\_contribution} \sqcap \neg\mathsf{Paper}\\ \mathsf{Paper} &\equiv \mathsf{Regular\_contribution} \sqcap \neg\mathsf{Extended\_abstract}\end{aligned}$$

(*#3-8*) Domain/role concept of a simple/synonym/inverse role.
A role is defined as a relation between a certain domain and range concept, thus in some cases a role name itself is sufficient to define its domain or range, i.e. a single explicit definition axiom can be constructed as follows:

(#3) Domain : $\mathsf{Wife} \equiv \exists.\mathsf{hasHusband}.\top$ *(#4) Range* : $\mathsf{Husband} \equiv \exists.\mathsf{hasHusband}^{-}.\top$

The next real ontology example illustrates the domain/role concept of a simple role patterns:

[8] CONFERENCE corpus, Conference.owl

Example 3.7. [9] Document *is definable either as the domain of* used_by*, or as the range of* use*.*

$$\begin{aligned}\mathcal{J} = \{\ &\alpha_1 : \mathsf{used\_by} \equiv \mathsf{use}^{-},\\ &\alpha_2 : \mathsf{Document} \sqsubseteq \exists \mathsf{used\_by}.\mathsf{Person}\\ &\alpha_3 : \top \sqsubseteq \forall \mathsf{use}.\mathsf{Document}\}\end{aligned}$$

Using the role used_by *as the concept definition signature, the axioms are rewritable entailing an explicit definition as follows:*

$$\alpha_3 \rightsquigarrow \top \sqsubseteq \forall \mathsf{usedBy}^{-}.\mathsf{Document} \quad \textit{thus} \quad \mathsf{Document} \equiv \exists \mathsf{used\_by}.\top$$

Alternatively, one may employ the role use *as the definition signature can express the defined concept as:*

$$\alpha_2 \rightsquigarrow \mathsf{Document} \sqsubseteq \exists \mathsf{use}^{-}.\mathsf{Person} \quad \textit{thus} \quad \mathsf{Document} \equiv \exists \mathsf{use}^{-}.\top$$

Note for definition extraction. The justification of such patterns where the defined concept is either the domain or the range of the single MDS member role (i.e. *#3-8*) could contain an arbitrary number and shape of axioms, which could make extraction difficult. Therefore instead of parsing the justification in order to derive the definition axiom, we propose testing whether the justification entails a domain or range axiom and selecting the one that is entailed.

(*#3*) Domain concept of a role.
Matching pattern

- Defined concept: C
- MDS: contains a single role name ($|\Sigma| = 1$) e.g. $\Sigma^{\mathsf{C}} = \{\mathsf{r}\}$
- Justification: contains more than one axiom ($|\mathcal{J}| \geq 2$), e.g. $\mathcal{J} = \{\mathsf{C} \sqsubseteq \exists \mathsf{r}.\top, \exists \mathsf{r}.\top \sqsubseteq \mathsf{C}\}$

Extraction pattern

- Definition: *Defined concept* is equivalent to the *domain of the single MDS member role name*, e.g. $\mathsf{C} \equiv \exists.\mathsf{r}.\top$

(*#4*) Range concept of a role.
Matching pattern

- Defined concept: C
- MDS: contains a single role name ($|\Sigma| = 1$) e.g. $\Sigma^{\mathsf{C}} = \{\mathsf{r}\}$
- Justification: contains more than one axiom ($|\mathcal{J}| \geq 2$), e.g. $\mathcal{J} = \{\top \sqsubseteq \forall \mathsf{r}.\mathsf{C}, \mathsf{C} \sqsubseteq \exists \mathsf{r}^{-}.\top\}$

[9] Conference corpus, *cocus.owl*

Extraction pattern

- Definition: *Defined concept* is equivalent to the *range of the single MDS member role name*, e.g. $\mathsf{C} \equiv \exists.\mathsf{r}^{-}.\top$

(#5) Domain concept of a synonym role.
Matching pattern

- Defined concept: C
- MDS: contains a single role name ($|\Sigma| = 1$) e.g. $\Sigma^{\mathsf{C}} = \{\mathsf{s}\}$
- Justification: contains more than one axiom, including a set of axioms that implies role synonymity ($|\mathcal{J}| \geq 2$), e.g. $\mathcal{J} = \{\mathsf{C} \sqsubseteq \exists\mathsf{r}.\top, \exists\mathsf{r}.\top \sqsubseteq \mathsf{C}, \mathsf{r} \equiv \mathsf{s}\}$

Extraction pattern

- Definition: *Defined concept* is equivalent to the *domain of the single MDS member synonym role name*, e.g. $\mathsf{C} \equiv \exists.\mathsf{s}.\top$

(#6) Range concept of a synonym role.
Matching pattern

- Defined concept: C
- MDS: contains a single role name ($|\Sigma| = 1$) e.g. $\Sigma^{\mathsf{C}} = \{\mathsf{s}\}$
- Justification: contains more than one axiom ($|\mathcal{J}| \geq 2$), including a set of axioms that implies role synonymity, e.g. $\mathcal{J} = \{\top \sqsubseteq \forall\mathsf{r}.\mathsf{C}, \mathsf{C} \sqsubseteq \exists\mathsf{r}^{-}.\top, \mathsf{r} \equiv \mathsf{s}\}$

Extraction pattern

- Definition: *Defined concept* is equivalent to the *range of the single MDS member synonym role name*, e.g. $\mathsf{C} \equiv \exists.\mathsf{s}^{-}.\top$

(#7) Domain concept of an inverse role.
Matching pattern

- Defined concept: C
- MDS: contains a single role name ($|\Sigma| = 1$) e.g. $\Sigma^{\mathsf{C}} = \{\mathsf{s}\}$
- Justification: contains more than one axiom, including a set of axioms that implies the inverse role relation ($|\mathcal{J}| \geq 2$), e.g. $\mathcal{J} = \{\mathsf{C} \sqsubseteq \exists\mathsf{r}.\top, \exists\mathsf{r}.\top \sqsubseteq \mathsf{C}, \mathsf{r} \equiv \mathsf{s}^{-}\}$

Extraction pattern

- Definition: *Defined concept* is equivalent to the *domain of the single MDS member role name*, e.g. $\mathsf{C} \equiv \exists.\mathsf{s}.\top$

(#8) Range concept of an inverse role.
Matching pattern

- Defined concept: C
- MDS: contains a single role name ($|\Sigma| = 1$) e.g. $\Sigma^{\mathsf{C}} = \{\mathsf{s}\}$
- Justification: contains more than one axiom, including a set of axioms that implies the inverse role relation, e.g. $\mathcal{J} = \{\top \sqsubseteq \forall \mathsf{r}.\mathsf{C}, \mathsf{C} \sqsubseteq \exists \mathsf{r}^{-}.\top, \mathsf{r} \equiv \mathsf{s}^{-}\}$

Extraction pattern

- Definition: *Defined concept* is equivalent to the *range of the single MDS member role name*, e.g. $\mathsf{C} \equiv \exists.\mathsf{s}^{-}.\top$

3.4.2 Role Definition Patterns

(#9) Implicit synonym role.
Matching pattern

- Defined role: r
- MDS: contains a single role name, which is the synonym of the defined role ($|\Sigma| = 1$) e.g. $\Sigma^{\mathsf{r}} = \{\mathsf{s}\}$
- Justification: contains more than one axiom ($|\mathcal{J}| \geq 2$), e.g. $\mathcal{J} = \{\mathsf{r} \sqsubseteq \mathsf{s}, \mathsf{s} \sqsubseteq \mathsf{r}\}$

Extraction pattern

- Definition: *Defined role* is equivalent to the *single MDS member role name*, e.g. $\mathsf{r} \equiv \mathsf{s}$

(#10) Implicit inverse role.
Matching pattern

- Defined role: r
- MDS: contains a single role name, which is the inverse of the defined role ($|\Sigma| = 1$) e.g. $\Sigma^{\mathsf{r}} = \{\mathsf{p}\}$
- Justification: contains more than one axiom ($|\mathcal{J}| \geq 2$), e.g. $\mathcal{J} = \{\mathsf{r} \equiv \mathsf{s}, \mathsf{s} \equiv \mathsf{p}^{-}\}$

Extraction pattern

- Definition: *Defined role* is equivalent to the *inverse of the single MDS member role name*, e.g. $\mathsf{r} \equiv \mathsf{p}^{-}$

3.5 Definability and Ontology Modelling

In addition to definition patterns, the empirical analysis (presented in Chapter 5) has also highlighted a direct application of the results of MDSs computation that can help identifying modelling errors in ontologies thus aid debugging. Note that case 1 and 2 can both be identified by mere classification (case 1 was discussed in literature). However if one tries to detect case 3, which requires computing MDSs, the other cases would be detected as well without carrying out an additional reasoning task (i.e. classification)

This section presents the formalisation of three types of errors, each of which can be automatically detected, but their repairs require the involvement of an ontology engineer and a domain expert.

(1) Implicitly defined by an empty DS. The only concepts in any ontology, which requires no signature for their definition are $\top$ and $\bot$. This error can also be detected by classification, or by querying the TBox for any named concept C such that $\mathcal{T} \models \mathsf{C} \equiv \top$ or $\mathcal{T} \models \mathsf{C} \equiv \bot$.

If a named concept is definable by an empty signature, then the ontology is most likely to contain an error. For example in the *cocus.owl* ontology[10] $\mathsf{Person} \equiv \top$. By examining the document, it becomes obvious that this is unintentional, as the ontology contains many other concepts (such as Conference) that are definitely not semantically related to Person.

Another possibility is this is not necessarily a modelling error as some ontologies intentionally introduce an explicit name for $\top$ to explicitly describe the domain of the ontology (e.g. the *Top level concept* in SNOMED CT [11]).

(2) Unwanted synonym(s). These occur when two or more concepts, meant to convey different meaning, are wrongly represented as interchangeable synonyms of one another. Figure 3.3 shows three different ways of defining the concept Anthropometrics_Height[12]. Obviously, Anthropometrics_Height, Anthropometrics_Weight and Anthropometrics_BMI are semantically related, but different concepts. However, in TBox $\mathcal{T}$ where $(\mathcal{J}_1 \cup \mathcal{J}_2 \cup \mathcal{J}_3) \subseteq \mathcal{T}$, these concepts are defined as equivalent. The correction requires expert knowledge. Anthropometrics means measurement of the size and proportions of the human body. Axioms $\alpha_2, \alpha_3, \alpha_4$ are correct, as *height, weight* and *BMI* are all type of measurements that make up the general class Anthropometrics, but axiom α_1 is incorrect as *height* and *weight* measurements would share nothing in common, i.e. their intersection would be empty. An appropriate representation would describe Anthropometrics as the disjoint union of these concepts.

[10] From OAEI 2014 Campaign Conference-track corpus. It is worth to note that these ontologies are curated by experts, thus such an error is unexpected.

[11] `https://confluence.ihtsdotools.org/display/DOCGLOSS/Top+level+concept`

[12] Bioportal corpus, bp26.owl

$$\mathcal{J}_1 = \{\alpha_1 : \mathsf{Anthropometrics} \sqsubseteq \mathsf{Anthropometrics\_BMI} \sqcap \mathsf{Anthropometrics\_Height} \sqcap \mathsf{Anthropometrics\_Weight},$$
$$\alpha_2 : \mathsf{Anthropometrics\_Height} \sqsubseteq \mathsf{Anthropometrics},$$
$$\alpha_3 : \mathsf{Anthropometrics\_Weight} \sqsubseteq \mathsf{Anthropometrics}\}$$
$$\models \boxed{\mathit{Anthropometrics\_Height} \equiv \mathsf{Anthropometrics\_Weight}}$$

$$\mathcal{J}_2 = \{\alpha_1 : \mathsf{Anthropometrics} \sqsubseteq \mathsf{Anthropometrics\_BMI} \sqcap \mathsf{Anthropometrics\_Height} \sqcap \mathsf{Anthropometrics\_Weight},$$
$$\alpha_2 : \mathsf{Anthropometrics\_Height} \sqsubseteq \mathsf{Anthropometrics}\}$$
$$\models \boxed{\mathit{Anthropometrics\_Height} \equiv \mathsf{Anthropometrics\_BMI}}$$

$$\mathcal{J}_3 = \{\alpha_1 : \mathsf{Anthropometrics} \sqsubseteq \mathsf{Anthropometrics\_BMI} \sqcap \mathsf{Anthropometrics\_Height} \sqcap \mathsf{Anthropometrics\_Weight},$$
$$\alpha_2 : \mathsf{Anthropometrics\_Height} \sqsubseteq \mathsf{Anthropometrics},$$
$$\alpha_4 : \mathsf{Anthropometrics\_BMI} \sqsubseteq \mathsf{Anthropometrics}\}$$
$$\models \boxed{\mathit{Anthropometrics\_Height} \equiv \mathsf{Anthropometrics}}$$

FIGURE 3.3: Unwanted synonyms modelling error: three concepts that should be different, are semantically equivalent to each other.

$$\mathcal{J} = \{\alpha_1 : \mathit{Regular\_author} \equiv (\mathsf{Contribution\_1th} - \mathsf{author} \sqcup \mathsf{Contribution\_co} - \mathsf{author}) \sqcap (\underbrace{\exists \mathsf{contributes}.\mathsf{Conference\_contribution}}_{\text{redundant}})$$
$$\alpha_2 : \mathsf{Contribution\_1th} - \mathsf{author} \sqsubseteq \mathsf{Regular\_author},$$
$$\alpha_3 : \mathsf{Contribution\_co} - \mathsf{author} \sqsubseteq \mathsf{Regular\_author}\}$$

$$\alpha_1 \models \mathsf{Regular\_author} \sqsubseteq (\mathsf{Contribution\_1th} - \mathsf{author} \sqcup \mathsf{Contribution\_co} - \mathsf{author})$$
$$\{\alpha_2, \alpha_3\} \models (\mathsf{Contribution\_1th} - \mathsf{author} \sqcup \mathsf{Contribution\_co} - \mathsf{author}) \sqsubseteq \mathsf{Regular\_author}$$

$$\mathcal{J} \models \mathit{Regular\_author} \equiv \mathsf{Contribution\_1th} - \mathsf{author} \sqcup \mathsf{Contribution\_co} - \mathsf{author}$$

FIGURE 3.4: Redundant concepts in explicit definition.

(3) Redundant concept(s). This is not necessarily a modelling error, but a discrepancy between the *intended meaning* (formalised by explicit definition axioms), and the *actual meaning* (the alternative explicit definition, which corresponds to an MDS of the defined concept). Figure 3.4 presents an example of this case[13], here Regular_author is defined by axiom α_1, however, its signature is not minimal, because its subset:

$$\{\mathsf{Contribution\_1th} - \mathsf{author}, \mathsf{Contribution\_co} - \mathsf{author}\}$$

can also be used to define the concept, as it is implied by the justification. If the concept definition would be replaced by a more succinct form as

$$\mathsf{Contribution\_1th} - \mathsf{author} \sqcup \mathsf{Contribution\_co} - \mathsf{author}$$

[13]Conference corpus, conference.owl

the original definition would lose

$$\sqcap\ (\exists \mathsf{contributes.Conference\_contribution})$$

which would be a real loss regarding the human cognition of the axiom.

This case occurs frequently, e.g. in *SNO_nci*, 76.28% of all MDSs had redundant concepts in explicit definitions. Therefore we assume that often knowledge engineers may add semantically redundant entities to certain definitions in order to aid human cognition, because ontologies are meant be meaningful and comprehensible for both humans and machines.

Discussion. Case 1 and 3 are not strictly modelling errors, i.e. these might occur due to intentional choices made during the design and implementation of an ontology. The application of MDSs informs users about these particular properties of axioms so they are aware and if it is necessary they may adjust the axiomatisation (e.g. remove concept equality to Thing, or make an axiom more succinct).

3.6 Discussion on the applications of MDSs and Beth definability

As previously discussed in Section 3.1, the Beth definability property means that whenever a given concept (or role) has an implicit definition it also should have an explicit definition. Implicit definability is a semantic property of ontologies, signatures, and concepts (or roles) stating that whenever the signature is fixed under a given ontology then the definiton of a particular concept (or role) is also fixed. One implication of Beth's result is that if that is the case, then in principle one can always generate an explicit definition for an implicitly definable concept (or role). This requires that the particular language, which has been used to formulate the ontology that contains the aformentioned concept or role, accepts the Beth definability property.

In this work we only focus on implicit definitions, i.e. make use of the fact that it suffices to know that fixing the values of definition signature symbols also fixes the values for the given concept (or role) that is implicitly definable by the signature. The motivation for this work is to improve the results of ontology matching by considering definability-based alignments. For example, let us consider two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ that are aligned to each other. Let C and C' be implicitly defined concepts under $\mathcal{O}_1$ and $\mathcal{O}_2$ using the definition signature Σ and Σ' respectively, where the symbols of Σ are mapped to Σ'. If the values of Σ symbols are fixed then it fixes the value of C under $\mathcal{O}_1$; and if the values of Σ' symbols (that correspond to mappings) are fixed then it fixes the value of C' under $\mathcal{O}_2$, which suggests that C and C' might be the same thing. This application scenario does not rely on explicit definitions.

Due to the fact that *(i) implicit definability check works in languages that do not accept Beth definability* and that *(ii) definability-based alignment relies on definition signatures not definitions themselves*, explicit definitions hence the Beth definability property is not required for the practical applications of this thesis. Therefore from now on in this work we only consider using implicit definability.

Note that in such application scenarios where explicit definitions are also needed, one need to make sure to use an ontology language for which the Beth definability property has been proven to hold.

3.7 Summary and Conclusions

- This chapter has introduced the *notion of explicit and implicit definability*, which is applicable to both concepts and roles. An *entity can be defined either explicitly, or implicitly* under an ontology, or otherwise it is undefined. Every implicitly defined entity is also explicitly definable (given that the Beth definability property holds for the particular ontology language). A *defined entity is rewritable into at least one, but potentially exponentially many* (in the size of the ontology) syntactically different and semantically equivalent form(s).

- *Definability can be detected* using the algorithms presented in this chapter. Detecting whether an entity is explicit defined is a trivial process, whereas deciding implicit definability is a more complex process that requires reasoning (i.e. entailment check, whose complexity varies from logic to logic and it may be exponential in the number of the axioms in the ontology).

- *Definition signatures (DS and minimal DS)* were presented to characterise implicitly definable entities in terms of their explicit definability. Signatures provide the underpinning definability, as presented in later chapters.

- *Since explicit definitions are not required for the practical applications of MDSs, and implicit definability check works in languages that do not accept Beth definability*, Beth definability is not required. Thus in this work we only consider using the notion of implicit definability.

- As definability is often difficult for humans to comprehend, the implicit definability check algorithm was modified to compute *justifications*, i.e. minimal sets of axioms that are sufficient for entailments to hold. In addition to validation, justifications can support (i.e. serve as input) heuristic-based axiom generation.

- *Some entity definition signatures and justifications correspond to patterns.* This chapter has described a non-comprehensive study on what are the most common definition patterns that have emerged during the empirical evaluation of MDS computation (in the particular set ontologies of the evaluation corpus), and presented

an incomplete list of definition patterns, that can be used for a heuristics-based axiom generation approach using matching and extraction patterns.

- *MDSs can help in identifying modelling errors in ontologies.* Three types of errors were formalised, each of which can be automatically detected, but their repairs requires the involvement of an ontology engineer, and potentially a domain expert.

Chapter 4

Computing Minimal Definition Signatures

This chapter introduces the problem of finding minimal definition signatures (MDSs), describes a naive and an optimised approach for obtaining MDSs, and provides the concrete details of the MDS computation *algorithms for concepts*. Algorithms for finding role MDSs are omitted as the methods presented are trivial to modify in order to support roles.

Please note that in all algorithms presented in this chapter we consider the reasoning task of deciding implicit definability, which is dedicated to an external reasoner system, as constant time single step computation, i.e. treating them as oracle calls [107]. Due to the fact that MDS computation is considered for an arbitrary DL logic language, where the complexity of the oracle call varies from logic to logic, and because the advancements of standard reasoning tools made them manageable for many DLs, each algorithm complexity analysis applies just to the complexity of the algorithm in question and not to the combined running time of the algorithm and the implicit definability check(s) performed by the oracle.

The remainder of this chapter is organised as follows: Section 4.1 examines the underlying computational complexity of the MDS search problem, and outlines the MDS computation process. Section 4.2 presents the structure of the MDS computation algorithms and shows how the various sub-routines are integrated together in order to form algorithms for finding a single MDS, a set of mutually disjoint MDSs, or finding all MDSs of defined concepts. Sections from 4.3 to 4.6 describe the various algorithms and optimisation procedures. Section 4.7 investigates how modularisation impacts the complexity of definability and MDS computation. Section 4.8 summarises and concludes the chapter.

4.1 The MDS Computation Process

Finding *a single MDSs* for a given concept (or role) is potentially a computationally expensive process:

- The *set of candidate signatures* for any given concept (or role) is the power set of the TBox signature (excluding the defined concept (or role) itself, i.e. the candidate signature set is $\mathcal{P}(\mathsf{Sig}(\mathcal{T}) \setminus \{\mathsf{C}\})$, therefore exhaustively testing all candidates requires $2^{|(\mathsf{Sig}(\mathcal{T})\setminus\{\mathsf{C}\})|}$ number of calls to the oracle (reasoner).

- Each candidate signature must be subjected to an *implicit definability check*, which is performed by the oracle, where the complexity of testing each signature is predicated on the DL expressivity of the given ontology language, thus it may take exponential time in the number of axioms in the ontology, for more expressive DL flavours.

Furthermore, finding the *complete set of MDSs* for all defined concepts (or roles) poses an even bigger challenge, as previously shown (Section 3.1), the *number of MDSs* of a given defined concept can be exponential in the size of the ontology signature. Despite the high complexity, there are several factors implying that the typical MDS computation is expected to stay within manageable bounds and thus feasible to compute:

- The implicit definability check is performed by a *reasoner* (which is dedicated to an oracle). In practice, state of the art systems are demonstrated to work for most of the currently existing TBoxes, even for more expressive DL languages.

- *Modularization* can be applied as a space reduction mechanism. The implicit definability check process can be optimised by operating on a module, instead of the entire TBox. Furthermore, the initial search space, $\mathcal{P}(\mathsf{Sig}(\mathcal{T}) \setminus \{C\})$, can be reduced in size by applying modularization. A module $\mathcal{M}^{\mathsf{C}}$, which describes a defined concept (where $\mathcal{M}^{\mathsf{C}} = \mathsf{Mod}(\{C\}, \mathcal{T})$) is usually significantly smaller than the ontology, thus the search space can be decreased to $\mathcal{P}(\mathsf{Sig}(\mathcal{M}^{\mathsf{C}}) \setminus \{C\})$.

- For most entities, the *number of MDSs is low*. Defined concepts (and roles) are described in terms of other entities, hence the number of MDSs for a given entity depends on the number of MDSs of its description entities. Therefore, in general, the number of "less-defined" entities (entities with relatively low number of MDSs) is expected to be considerably larger than the "more-defined" ones.

Definability computation builds on the implicit definability check method (Section 3.3.2). Hence the basic idea behind obtaining MDSs is that by reducing the number of candidate signatures required to be tested, the overall complexity can be decreased. Figure 4.1 illustrates the definability computation process, which consists of three main stages:

(*1*) The first stage establishes the *definability status* of a given entity, using the optimised approach presented in Section 3.3.2; that first test for explicit definitions, before proceeding with the implicit definability check (i.e. making one call to the oracle in the number of axioms in the ontology where the oracle call may itself take exponential time in the number of axioms in the ontology). However, prior to

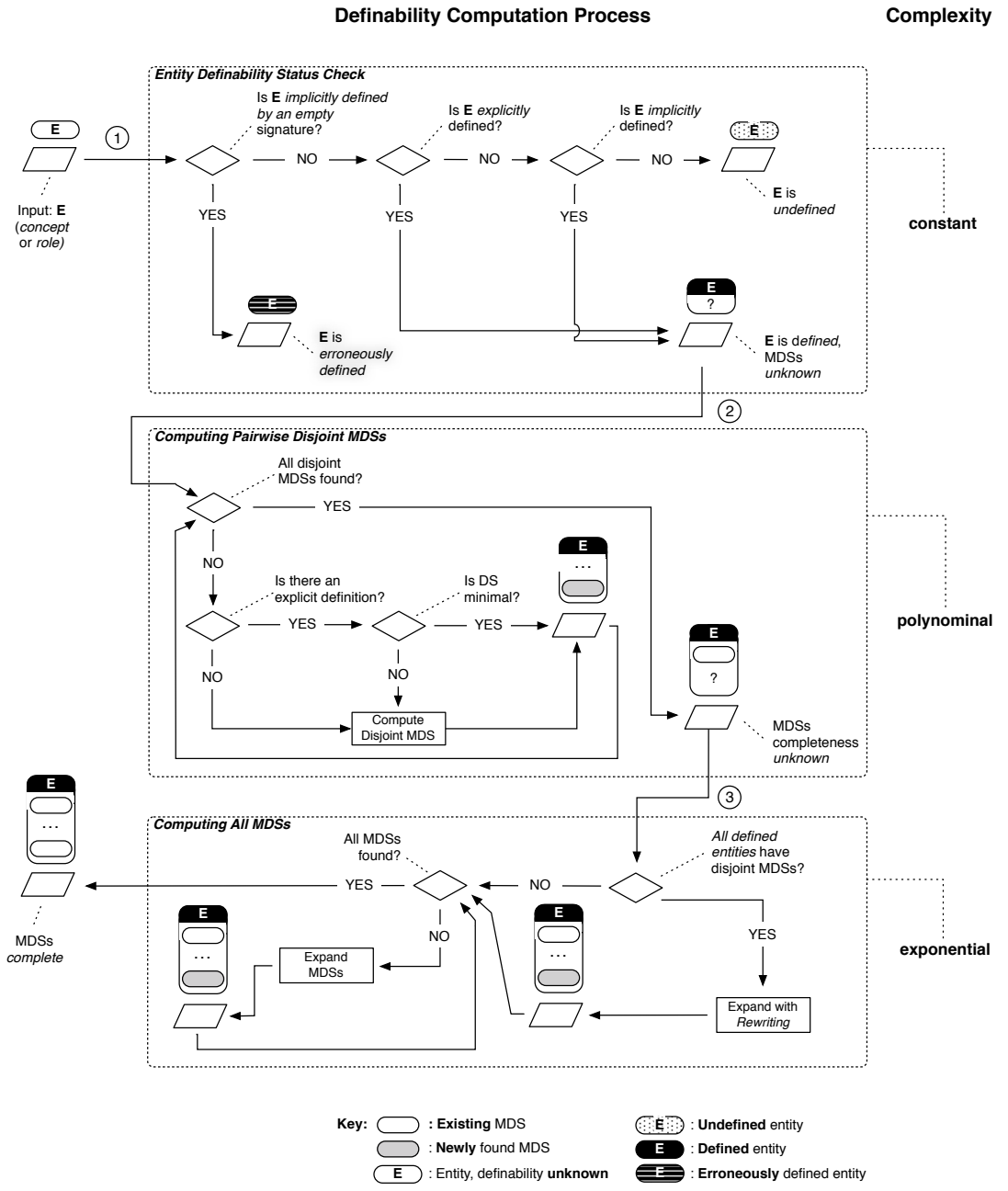


FIGURE 4.1: The definability computation process.

checking the actual definability status of a given entity, it is important to rule out a possible modelling error. This is obtained by checking if our entity e is implicitly definable by an empty signature (i.e. $\mathcal{T} \models \mathsf{E} \equiv \top$ or $\mathcal{T} \models \mathsf{E} \equiv \bot$). In this case it is not necessary to determine MDSs as these entities are rewritable by $\top$ or $\bot$. The overall complexity of this stage is given by two oracle calls (testing implicit definability of an entity with an empty signature, and the whole ontology signature excluding the entity in question), and the simple explicit definability check.

(*2*) The second stage obtains a set of pairwise disjoint *disjoint MDSs*. Each defined entity has at least one, but potentially exponentially many (i.e. $2^{|(\mathsf{Sig}(\mathcal{T})\setminus\{\mathsf{E}\})|}$),

unique MDSs in an ontology. However the number of disjoint MDSs for a given entity E is at most n, where $n = |\mathsf{Sig}(\mathcal{T}) \setminus \{\mathsf{E}\}|$ (the smallest possible valid MDS contains one entity); thus it is linear in the size of the ontology signature. Every defined entity is known to be defined by the DS: $\mathsf{Sig}(\mathcal{T}) \setminus \{\mathsf{E}\}$. In order to find a set of pairwise disjoint signatures, the process iteratively reduces this *working signature* with one extracted MDS at the time, until the signature no longer implicitly defines the given entity. In some cases there could be multiple possible different MDS sets whose constituent MDSs are pairwise disjoint, thus the result of this stage may depend on the entity ordering in the initial signature. The stage is optimised for explicitly defined entities by first testing the signature of explicit definition axioms. Once each explicit definition signature is assessed for minimality (and if necessary reduced to an MDS), the process then finds any remaining pairwise disjoint MDSs.

(*3*) The final stage computes any potentially unidentified MDSs by expanding existing ones. While the first two stages make polynomial number of calls to the oracle in the number of axioms in the ontology where each oracle call may itself take exponential time in the number of axioms in the ontology, the third stage iterates through an exponential number of candidate signatures, making one oracle call for each candidate signature. However the number of candidate signatures, which is the power set of the union of existing MDSs, is typically low for most entities; furthermore, an increasing number of candidate signatures are excluded from testing during this process, based on the MDSs identified until that point. The stage can be further optimised when definability computation targets a set of entities, e.g. the ontology signature: prior to the expansion first all the disjoint MDSs are computed for each entity, then an expansion by rewriting strategy is applied to obtain more MDSs.

4.2 Algorithm Structure

Figure 4.2 presents a schematic of the various definability computation algorithms. All MDS computing algorithms are built upon performing some form of ***definability*** check (introduced in Chapter 3.3). IsConceptDefined (Algorithm 3) combines the implicit definability (Algorithm 2) and the explicit defintion (Algorithm 1) check to provide an optimised way of determining the definability status of a given entity.

This chapter presents two approaches that are able to find the *complete set of MDSs* of a defined entity:

- ComputeAllMDSs-BF (Algorithm 5) is a naive, brute-force approach that makes exponentially many number of calls to the oracle, as it exhaustively examines each subset of the power-set of the TBox signature. This method can only be used with tiny ontologies (containing less than twenty entities).

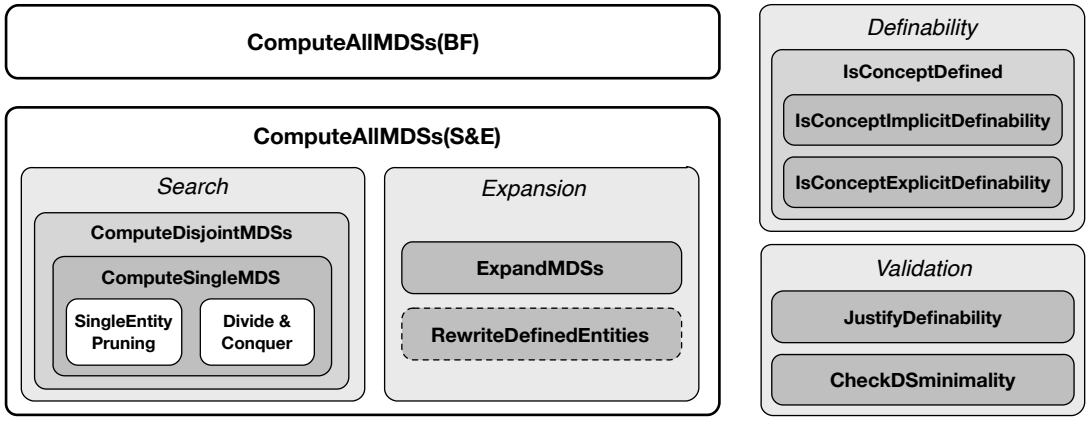


FIGURE 4.2: Definability (status and MDS) computation algorithm structure.

- COMPUTEALLMDSS-S&E (Algorithm 13) is an algorithm that combines a number of sub-routines, forming a method which also finds all MDSs, however typically significantly faster than the brute-force approach.

As shown in the algorithm structure diagram, COMPUTEALLMDSS-S&E consists of two parts, corresponding to the main execution phases: search and expansion. The ***search*** phase finds a set of pairwise disjoint MDSs making polynomial number of calls to the oracle. This phase employs the following algorithms:

- COMPUTESINGLEMDS (Section 4.4) this is the heart of the search phase. It is used to compute one MDS. It has two variants: SINGLEENTITYPRUNING (Algorithm 6), and DIVIDE&CONQUER (Algorithm 7). Although the latter version is generally faster, the first approach is also useful as it performs better in certain scenarios that fall into the worst case complexity of Algorithm 7.

- COMPUTEDISJOINTMDSS (Algorithm 10) obtains a set of mutually disjoint MDSs of a defined entity by repeatedly computing a single MDS, using either single-, or multi-entity pruning. Algorithm 11 provides an optimisation by first processing explicit definition signatures, thus the resulting MDSs are not always all mutually disjoint.

The potentially incomplete set of MDSs is then completed in the ***expansion*** phase, i.e. the final stage of definability computation. The phase consists of a mandatory algorithm and an optional algorithm, where both algorithms require a set of precomputed MDSs as input:

- EXPANDMDSS (Algorithm 12) evaluates whether all MDSs of a given defined entity have been found, by either obtaining new MDSs, or confirming that all possible MDSs have been found. This algorithm uses exponential number of calls (size of the power set of the union of a set of given set of MDSs) to the oracle in the number of axioms in the ontology where each oracle call may itself take exponential time in the number of axioms in the ontology.

- RewriteDefinedEnities (Algorithm 14) potentially finds new MDSs by replacing defined entities with their corresponding description signatures in existing MDSs. Although the process is not guaranteed to find new MDSs, it may optimise expansion in certain scenarios (e.g. when MDSs are not precomputed, but required in a dynamic environment) due to making polynomial number of calls to the oracle in the number of axioms in the ontology where each oracle call may itself take exponential time in the number of axioms in the ontology. Unlike other definability computation methods, this expansion approach utilises the set of existing MDSs of all defined entities that have been computed.

The minimal definition signature ***validitation*** is supported by the following algorithms:

- CheckDSMinimality (Algorithm 9) assesses the minimality of a definition signature by attempting to reduce the input DS into an MDS.
- JustifyDefinability (Algorithm 4) computes a justification (set of axioms), that aids human comprehension of definability, as well as it facilitates a heuristic-based definition axioms generation (Chapter 3.4).

The algorithms are presented as follows: Section 4.3 introduces the naive approach to finding all MDSs. Section 4.4 explains the algorithms for computing a single MDS, and the minimality validation method. Section 4.5 and Section 4.6.1 describe the algorithms that support the search and the expansion phase, respectively.

4.2.1 Algorithm Complexity Summary

Table 4.1 summarizes the complexity of the definability check (presented in Chapter 3) and MDS computation (presented in this Chapter) algorithms, where the left column

ID	Algorithm Signature	Nb of calls to the oracle
1	IsConceptExplicitlyDefined	0
2	IsConceptImplicitlyDefined	1
3	IsConceptDefined	at best:0, at worst:1
4	JustifyDefinability	1
5	ComputeAllMDS	at worst: $2^{\lvert\mathsf{Sig}(\mathcal{T})\rvert-1}$, i.e. exponential
6	ComputeSingleMDS	always n
7	ComputeSingleMDS-D&C	at best: $2 \cdot (\lfloor log_2 n \rfloor)$, at worst: $(2 * n) - 2$
9	CheckDSMinimality	at best: $2 \cdot (\lfloor log_2 n \rfloor)$, at worst: $(2 * n) - 2$
10	ComputeDisjointMDSs	at best: 2, at worst: $n + 1$
11	ComputeDisjointMDSs2	polynomial
12	ExpandMDSs	at worst $2^{\lvert\mathsf{Sig}(\mathcal{T})\setminus(\bigcup_{i=1}^{\lvert M\rvert} \forall\Sigma_i\{\Sigma_i\in M \mid 1<\lvert\Sigma_i\rvert\})\rvert}$ where M is the set of already identified MDSs of C
13	ComputeAllMDSs-S&E	at worst exponential
14	RewriteDefinedEntities	at worst exponential

Table 4.1: Complexity summary of MDS computation algorithms

show the algorithm ID, the middle column shows the algorithm name, and the right column describes the computational complexity of the algorithms in terms of the *number of calls to the oracle* in the number of axioms in the ontology where each oracle call may itself take exponential time in the number of axioms in the ontology (where $n = |\mathcal{S}|$ i.e. the number of entities in the input signature).

4.3 Computing All MDSs (Brute-force)

This algorithm formalises the first, naive approach to finding all MDSs of a given defined concept. This approach is is guaranteed to find all MDSs as it performs an exhaustive search through the set of all candidate signatures, i.e. the power-set of the TBox signature, excluding the defined concept. However as the algorithm uses exponential number of calls to the oracle in the number of axioms in the ontology where each oracle call may itself take exponential time in the number of axioms in the ontology, the practical application of this naive approach is limited only to tiny ontologies, thus in most cases it is unfeasible to use.

Algorithm 5: ComputeAllMDS($\mathsf{C}, \mathcal{T}$)

Input : C: defined concept; $\mathcal{T}$: TBox
Output: $M = \{\Sigma_1, \ldots, \Sigma_n\}$: the complete set of MDSs of C

```
1  M ← ∅
2  P ← PowerSet(Sig(T \ {C}))
3  for each S ∈ P do
4      if ∀Σ{Σ ∈ M|Σ ⊈ S|S ∈ P} then
5          if IsConceptImplicitlyDefined(C, T, S) then
6              M ← M ∪ {S}
7          end
8      end
9  end
10 return M
```

Walkthrough. The process first initialises the set M which holds the identified MDSs (line 1), and the search-space ($\mathcal{P}$) which contains all subsets of the ontology signature in ascending order with regards to candidate set cardinality (line 2). Next the algorithm begins to exhaustively iterate through each *candidate concept definition signature* of C, i.e. $\mathsf{S} \in \mathcal{P}$ (line 4). In order to ensure that S is minimal, it is compared with all the already identified MDSs (line 5). If any $\Sigma \in M$ is a subset of S, then S is clearly not minimal, therefore the next step, the implicit definability check, is skipped. Otherwise S is tested to see whether it implicit defines C under $\mathcal{T}$ (line 6). Each minimal and correct S is an MDS of C, thus it stored in Σ (line 7). The algorithm terminates when all candidate signatures are explored (line 10), and returns M, the resulting set of all MDSs of C (line 11).

Correctness. The algorithm operates under the assumption that C is implicitly defined by the signature $\mathsf{Sig}(\mathcal{T}) \setminus \{\mathsf{C}\}$ under $\mathcal{T}$, therefore there is at least one valid MDS to be found. Any obtained signature is a valid MDS of C. A signature is minimal if it contains no superfluous entities: the minimality property is assured by exploring the search space in an ascending order, and that each candidate is compared with all identified MDSs as a non-minimal definition signature is a superset of an MDS that was already found. MDS correctness is assessed by the implicit definability check method (Algorithm 2), which is proven to be correct [135].

Termination and Completness. The search space $\mathcal{P}$ contains every possible MDS of C, because it includes all combination of every entity that can be used to define C. Furthermore, the algorithm exhaustively searches through a *finite* search space, examining each candidate signature exactly once, thus it terminates after finding all MDSs.

Complexity. The algorithm uses an exponential number of calls to the oracle in the number of axioms in the ontology where each oracle call may itself take exponential time in the number of axioms in the ontology as the number of candidate signatures tested for definability during the process is at worst $2^{|\mathsf{Sig}(\mathcal{T})|-1}$. The process is optimised as the search space is effectively pruned by excluding every superset of each computed MDS. However the actual reduction in complexity depends on the number and size of the MDSs: smaller MDSs are subsets of more candidate signatures that can be excluded due to non-minimality.

4.4 Computing A Single MDS

This section describes two algorithms that can identify *one minimal definition signature* of a given defined concept. The first approach (Section 4.4.1), which makes a *polynominal* (i.e. the size of the candidate signature) number of calls to the oracle in the number of axioms in the ontology where each oracle call may itself take exponential time in the number of axioms in the ontology, performs single entity pruning in order to reduce an input definition signature into a minimal definition signature. The second approach (Section 4.4.2) is a considerably faster as it uses a divide and conquer strategy to prune the input signature by entity groups, instead of individual entities. Furthermore, Section 4.4.3 presents a method, which determines DS minimality based on the single entity pruning approach.

4.4.1 Single Entity Pruning

Algorithm 6 computes one MDS of a given defined concept, which is contained within the input signature $\mathcal{S}$, assuming that $\mathcal{S}$ implicitly defines the concept under the ontology.

Walkthrough. The basic idea behind the algorithm is to systematically prune the

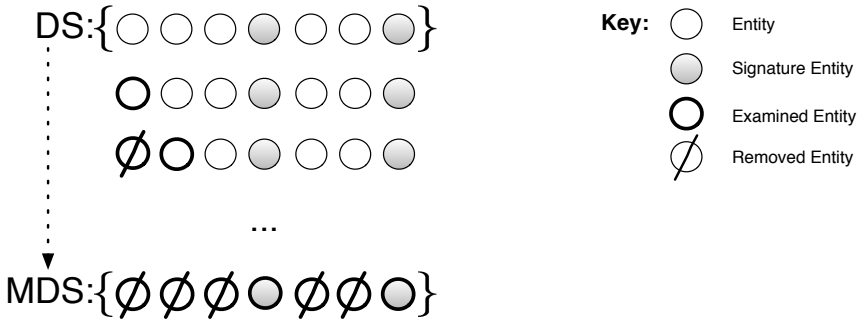


FIGURE 4.3: Computing an MDS from a DS with the single entity pruning approach.

Algorithm 6: COMPUTESINGLEMDS($\mathsf{C}, \mathcal{T}, \mathcal{S}$)

Input : C: defined concept; $\mathcal{T}$: TBox; $\mathcal{S}$: input signature; where $\mathsf{C}, \mathcal{S} \subseteq \mathsf{Sig}(\mathcal{T})$.
Output: Σ: one MDS of concept C

```
1 Σ ← S
2 for e ∈ Σ do
3 |   Σ ← Σ \ {e}
4 |   if IsConceptImplicitlyDefined(C, T, Σ) is False then
5 |   |   Σ ← Σ ∪ {e}
6 |   end
7 end
8 return Σ
```

input signature $\mathcal{S}$ until it contains no redundant members, thus it becomes *minimal*, while still implicitly defining concept C under $\mathcal{T}$. Figure 4.3 illustrates this process. The *prospective* minimal signature Σ is initialised with members of $\mathcal{S}$ (line 1). Pruning is achieved by removing a member of Σ (line 3) and testing the remaining signature to check if it still implicitly defines the given concept (line 4). If so, then the entity is redundant as opposed to being required; required entities are put back in Σ (line 5). The process is repeated until each signature member $\mathsf{e} \in \Sigma$ has been examined and Σ is minimal DS of C (line 2-7). If the input signature $\mathcal{S}$ contains more than one MDSs, then the outcome (i.e. the resulting one MDS) depends on the ordering of the entities in $\mathcal{S}$.

Correctness and Termination. As a precondition, $\mathcal{S}$ (and subsequently Σ) is assumed to be a valid definition signature of C. The process examines each member $\mathsf{e} \in \Sigma$ of the input signature exactly once, and removes only redundant entities, i.e. those that without Σ can still be used to describe C ($\mathcal{T} \models \mathsf{C} \equiv \mathsf{D}$, where $\mathsf{Sig}(\mathsf{D}) \subseteq \Sigma$). Therefore the output Σ contains no superfluous entities, thus the approach is correct. Furthermore it terminates when all elements of the (finite) input are exhausted.

Complexity. The process takes always n steps to complete (where n is the number of entities found in the input signature, i.e. $n = |\mathcal{S}|$), regardless of the number of required

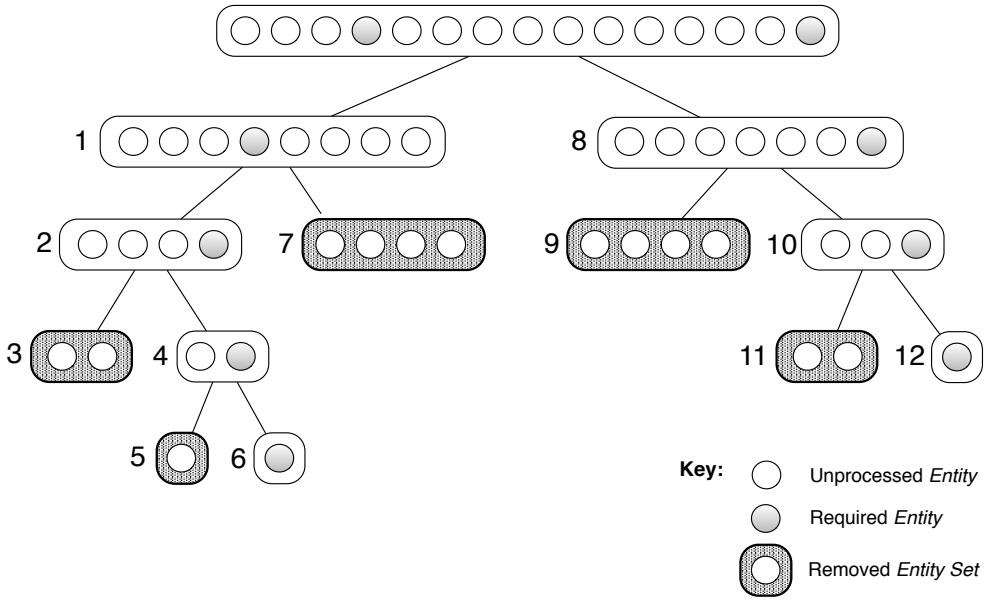


FIGURE 4.4: Computing an MDS using the divide and conquer approach. Tree nodes are labelled according to the order of traversal.

signature entities, because each entity is examined exactly once. Thus the algorithm makes a linear (n) number of calls to the oracle in the number of axioms in the ontology where each oracle call may itself take exponential time in the number of axioms in the ontology.

4.4.2 Divide and Conquer

Although the single entity pruning approach (Algorithm 6) makes polynomial number of calls to the oracle, for large signatures the process can still take considerable time because every entity is individually examined by performing an implicit definability check. Thus in order to reduce the overall complexity, we decrease the number of required implicit definability checks by employing a *divide and conquer strategy*.

The divide and conquer algorithm is commonly used in computer science [25]. The approach solves difficult problems by recursively splitting them into sub-problems until each part becomes simple enough to be solved. The final solution is derived by combining the results of sub-problems. Some applications of this approach include sorting (e.g. merge sort), searching (e.g. binary search) and syntactic analysis (e.g. top-down parsers).

The strategy is applied to MDS-search in the following way: instead of examining candidate signature entities individually, entities are tested in groups to determine whether they are required members of the MDS. Each entity group is recursively split until a smaller subset is found, such that it is either *removable* or cannot be split any further, i.e. contains only *required* MDS members. The process generates a *binary-tree*, where the root node is the input signature, and every other node is a subset of the input signature. Each leaf node consists of a set of entities that are either all removable, or all

Algorithm 7: ComputeSingleMDS-D&C($\mathsf{C}, \mathcal{T}, \mathcal{S}$)

Input : C: defined concept; $\mathcal{T}$: TBox; $\mathcal{S}$: input signature
Output: Σ: one minimal definition signature of concept C

1 $\mathcal{S}' \leftarrow \mathcal{S}$
2 $\mathbf{R} \leftarrow \emptyset$
3 $\mathbf{R} \leftarrow$ SplitAndPrune($\mathsf{C}, \mathcal{T}, \mathcal{S}', \mathcal{S}, \mathbf{R}$)
4 $\Sigma \leftarrow \mathcal{S} \setminus \mathbf{R}$
5 **return** Σ

Algorithm 8: SplitAndPrune($\mathsf{C}, \mathcal{T}, \mathcal{S}', \mathcal{S}, \mathbf{R}$)

Input : C: defined concept; $\mathcal{T}$: TBox; $\mathcal{S}'$: examined signature part; $\mathcal{S}$: original signature; $\mathbf{R}$: removable entities

1 **if** $|\mathcal{S}'| > 1$ **then**
2 $\quad \mathcal{S}_L, \mathcal{S}_R \leftarrow$ Split($\mathcal{S}'$)
3 $\quad \mathcal{S}_{check} \leftarrow \mathcal{S} \setminus (\mathbf{R} \cup \mathcal{S}_L)$
4 $\quad$ **if** IsConceptImplicitlyDefined($\mathsf{C}, \mathcal{T}, \mathcal{S}_{check}$) *is True* **then**
5 $\quad\quad \mathbf{R} \leftarrow \mathbf{R} \cup \mathcal{S}_L$
6 $\quad$ **end**
7 $\quad$ **else**
8 $\quad\quad \mathbf{R} \leftarrow$ SplitAndPrune($\mathsf{C}, \mathcal{T}, \mathcal{S}_L, \mathcal{S}, \mathbf{R}$)
9 $\quad$ **end**
10 $\quad \mathcal{S}_{check} \leftarrow \mathcal{S} \setminus (\mathbf{R} \cup \mathcal{S}_R)$
11 $\quad$ **if** IsConceptImplicitlyDefined($\mathsf{C}, \mathcal{T}, \mathcal{S}_{check}$) *is True* **then**
12 $\quad\quad \mathbf{R} \leftarrow \mathbf{R} \cup \mathcal{S}_R$
13 $\quad$ **end**
14 $\quad$ **else**
15 $\quad\quad \mathbf{R} \leftarrow$ SplitAndPrune($\mathsf{C}, \mathcal{T}, \mathcal{S}_R, \mathcal{S}, \mathbf{R}$)
16 $\quad$ **end**
17 **end**
18 **return** $\mathbf{R}$

required members of the MDS. The method is depicted by Figure 4.4 and formalised in Algorithm 7 & 8.

Walkthrough. The ComputeSingleMDS-D&C (Algorithm 7) serves as a main routine for its recursive subroutine SplitAndPrune (Algorithm 8). After initialisation (line 1-2) the runner calls SplitAndPrune (line 3) to identify $\mathbf{R}$, the set of redundant members in $\mathcal{S}$ (where $\mathbf{R} \subseteq \mathcal{S}$). The initial signature $\mathcal{S}$ is then pruned by removing $\mathbf{R}$ (line 4); the resulting signature Σ is a minimal DS of C, which is then returned and the process terminates (line 5).

SplitAndPrune takes an input signature $\mathcal{S}'$ (where $\mathcal{S}' \subseteq \mathcal{S}$, such that $\mathcal{S}$ denotes the initial signature provided by the main routine) and splits it into two parts $\mathcal{S}_L$ and $\mathcal{S}_R$ (line 2). Next it generates $\mathcal{S}_{check}$ (line 3) by taking the original signature $\mathcal{S}$ and removing: (1) $\mathbf{R}$, i.e. the set of all redundant entities found so far; (2) $\mathcal{S}_L$, i.e. one part of the currently examined signature. $\mathcal{S}_{check}$ is then tested to see whether it is a valid

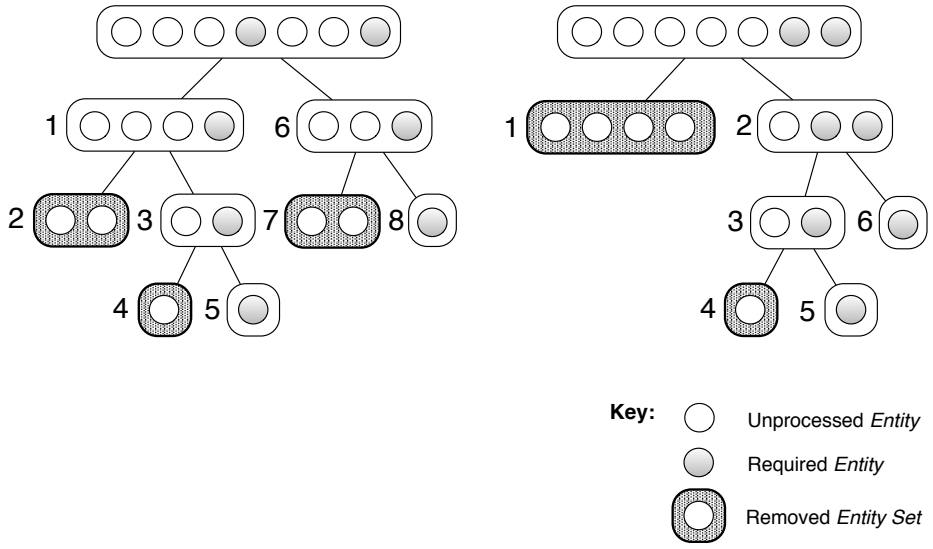


FIGURE 4.5: Computing MDSs with divide and conquer approach for two different signatures, with the same number of required (2 entities) and redundant members (6 entities). However, due to the ordering, the process takes less time to complete for the left-hand side signature.

definition signature (line 4). If so, then every member of $\mathcal{S}_L$ were *redundant entities*, thus these are added to $\mathbf{R}$ (line 5). However, if $\mathcal{S}_{check}$ fails the implicit definability check then the set $\mathcal{S}_L$ must contain at least one *required* entity, hence it cannot be removed as a whole, thus, recursively, SPLITANDPRUNE is called to identify redundant members of $\mathcal{S}_L$. Once any redundant member of $\mathcal{S}_L$ has been identified, the process is repeated for the other signature part, $\mathcal{S}_L$ (line 10-16). Finally the algorithm returns the redundant members of the input signature $\mathcal{S}'$, and terminates (line 18).

Complexity. The number of oracle calls depends on the ratio of required and redundant members in the signature, and the ordering of the signature. Figure 4.5 demonstrates how different signature orderings affect the process: both example input signatures contain the same number of redundant and required entities, however, due to the different entity ordering of these signatures, the MDS computation takes 8 steps for the left, and 6 steps for the right signature. The *worst case* occurs when all members $\mathsf{e} \in \mathcal{S}$ are required MDS entities, which is always unknown prior to computation. The required number of steps are $(2 * n) - 2$, where $n = |\mathcal{S}|$, i.e. the process makes linear number of calls to the oracle. In the *best case*, when the signature contains only one required entity, the number of required oracle calls is $2 \cdot (\lfloor log_2 n \rfloor)$, where $n = |\mathcal{S}|$ for all $|\mathcal{S}| \geq 4$. For instance, given a signature $|\mathcal{S}| = 1024$ with a single required member, it takes only 20 implicit definability checks to identify the MDS member, whereas the single entity pruning approach always requires $|\mathcal{S}|$ steps to process definition signatures.

Algorithm 9: CheckDSminimality($\mathsf{C}, \mathcal{T}, \mathcal{S}$)

Input : C: defined concept; $\mathcal{T}$: TBox; $\mathcal{S}$: definition signature of C that is being assessed for minimality

Output: the produced MDS Σ (which is equivalent to $\mathcal{S}$, if $\mathcal{S}$ was minimal)

```
1 if IsExplicitDefinitionSignature(C, S) then
2 |  Σ ← ComputeSingleMDS(C, T, S)
3 end
4 else
5 |  Σ ← ComputeSingleMDS-D&C(C, T, S)
6 end
7 return Σ
```

4.4.3 Determining Definition Signature Minimality

In general the number of required members of a candidate signature (i.e. members of the resulting MDS) is much smaller than the number of redundant entities. Therefore, typically, the divide and conquer method is more efficient to use. However, in some cases single entity pruning is better suited, in particular, when the signature which is tested for minimality is a signature of an explicit definition, as this often entails that the majority of signature members are required, i.e. such cases may reach the worst-case complexity of the divide and conquer approach. Algorithm 9 exploits this notion to provide an optimised way of determining signature minimality by attempting to reduce the input signature, using either the single (line 2), or the multi entity pruning approach, if the input does not correspond to any explicit definition signature (line 5); thus the complexity of this algorithm is given by the worst case of the divide and conquer approach. The returned minimal signature Σ is identical to $\mathcal{S}$ if the input was indeed minimal, otherwise $\mathcal{S}$ contains some redundant members thus the produced minimal DS is returned by the algorithm (line 7).

4.5 Computing Pairwise Disjoint MDSs

Algorithm 10 obtains *a set of mutually disjoint MDSs* of a given concept, assuming that the input signature implicitly defines the concept. In some cases the input signature may contain multiple possible different MDS sets that are pairwise disjoint, thus the result of the process may depend on the entity ordering of the input signature. Figure 4.6 depicts the approach.

Walkthrough. The basic idea behind the approach is that at every iteration (line 2-6), Σ, a single MDS of a defined concept C is computed (line 3) and subsequently (line 4) removed from the input signature $\mathcal{S}$, until $\mathcal{S}$ contains no unidentified MDSs, i.e. it no longer implicitly defines C under a TBox $\mathcal{T}$.

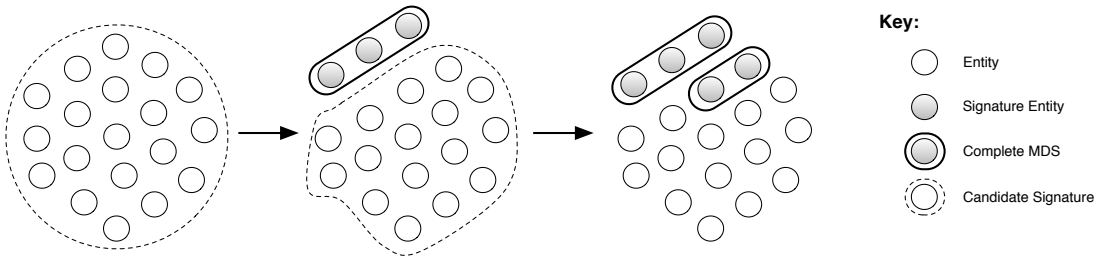


FIGURE 4.6: Computing mutually disjoint MDSs. Each iteration reduces the candidate signature by the last found MDS.

Algorithm 10: COMPUTEDISJOINTMDSS($\mathsf{C}, \mathcal{T}, \mathcal{S}$)

Input : C: defined concept; $\mathcal{T}$: TBox; $\mathcal{S}$: definition signature
Output: $M = \{\Sigma_1, \ldots, \Sigma_n\}$: pairwise disjoint MDSs of C

1 $M \leftarrow \emptyset$
2 **while** ISCONCEPTIMPLICITLYDEFINED($\mathsf{C}, \mathcal{T}, \mathcal{S}$) *is True* **do**
3 $\quad \Sigma \leftarrow$ COMPUTESINGLEMDS-D&C($\mathsf{C}, \mathcal{T}, \mathcal{S}$)
4 $\quad \mathcal{S} \leftarrow \mathcal{S} \setminus \Sigma$
5 $\quad M \leftarrow M \cup \{\Sigma\}$
6 **end**
7 **return** M

Correctness & Completeness. The correctness of this algorithm is determined by the following conditions. Let $\mathcal{S}_{start}$ denote the state of $\mathcal{S}$ prior to entering the main loop (at input), and $\mathcal{S}_{end}$ denote the state after completing the loop (line 7). The precondition of the process is that C is implicitly defined by $\mathcal{S}_{start}$ under $\mathcal{T}$. The postconditions of the process are: (*i*) C is definable by $\mathcal{S}_{end}$; (*ii*) $\mathcal{S}_{start} = \mathcal{S}_{end} \cup (\Sigma_1 \cup \ldots \cup \Sigma_n)$, where $n = |M|$; (*iii*) all MDSs are disjoint, i.e. $\forall \Sigma \{\Sigma \in M | \Sigma_i \cap \Sigma_j \neq \emptyset,$ where $i \neq j, 1 \leq i, j \leq n\}$. As each pairwise disjoint MDS is computed by using Algorithm 6 (that was shown to be correct) this algorithm is correct as well. Completeness is ensured by the postconditions. Moreover, the algorithm finds a set of MDSs, that are pairwise disjoint, however the result depends on the particular ordering of the input signature members; i.e. the same input can yield different solutions by changing the ordering, however the algorithm is deterministic w.r.t. the order of entities in $\mathcal{S}$.

Termination & Complexity. The algorithm operates on the working signature $\mathcal{S}$, which is a finite set of entities. At each step, $\mathcal{S}$ is pruned by removing entities of the MDS found in the previous step. Once there are no more remaining MDSs, the working signature set $\mathcal{S}$ fails the implicit definability check, thus the process terminates. Every MDSs is an entity set containing at least one member, thus at most the input contains $|\mathcal{S}|$ number of disjoint MDSs. Therefore the algorithm makes a uses a polynomial number of calls to the oracle in the number of axioms in the ontology where each oracle call may itself take exponential time in the number of axioms in the ontology. The actual number of oracle calls is $|M| + 1$, i.e. the number of identifiable mutually disjoint MDSs, and a

Algorithm 11: ComputeDisjointMDSs2($\mathsf{C}, \mathcal{T}$)

Input : C: defined concept; $\mathcal{T}$: TBox
Output: $M = \{\Sigma_1, \ldots, \Sigma_n\}$: not necessarily all pairwise disjoint MDSs of C

```
1  M ← ∅
2  S ← (Sig(T) \ C)
3  if IsConceptDefined(C, T) = Explicitly then
4  |   Axs ← GetExplicitDefinitionAxioms(C, T)
5  |   for α ∈ Axs do
6  |   |   Σ ← ComputeSingleMDS(C, T, (Sig(α) \ {C}))
7  |   |   M ← M ∪ {Σ}
8  |   |   S \ Σ
9  |   end
10 end
11 M ← M ∪ ComputeDisjointMDSs(C, T, S)
12 return Σ
```

last step which terminates the process. At the best case there is exactly one MDS, at the worst case there are $|\mathcal{S}|$ number of disjoint MDSs in the input signature.

4.5.1 Optimisation

In case the concept for which we are computing disjoint MDSs, is *explicitly defined*, the process can be optimised by first computing MDSs from the explicit definition axioms. This method is formalised by Algorithm 11. First the process gathers all explicit definition axioms of C (line 4), then it computes and stores an MDS from each axiom signature (line 6-7), and it subsequently prunes the working signature $\mathcal{S}$ by removing members of the MDS (line 8). Finally, any remaining pairwise disjoint MDSs are extracted from the potentially reduced input signature, using Algorithm 10 (line 11). As a given defined concept may have multiple explicit definitions whose signatures could overlap, the MDSs computed by this approach are not always pairwise disjoint. Moreover, this approach is focused on finding MDSs in the entire ontology, whereas the previous version facilitates extracting disjoint MDSs from a given signature.

4.6 Computing All MDSs

In order to finalise the potentially incomplete set of MDSs of a given defined entity, the last stage of the definability computation process attempts to expand upon existing MDSs to either identify new MDSs, or to confirm that all MDSs have been found. Section 4.6.1 presents the expansion algorithm, which despite the exponential number of calls made to the oracle, may support obtaining MDSs in practice. Section 4.6.2 provides the main algorithm which integrates several of the algorithms discussed in this chapter, in order find the complete set of MDSs of a defined entity, by performing the search and expand strategy. Furthermore, Section 4.6.3 describes a potential optimisation of the expansion phase.

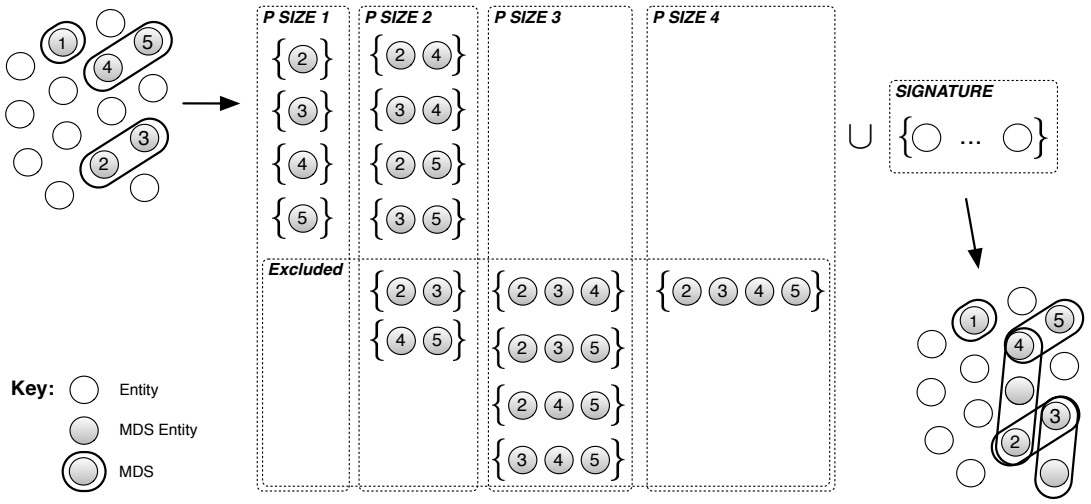


FIGURE 4.7: Expanding existing MDSs by combining and testing them with other non-signature entities.

Algorithm 12: EXPANDMDSS($\mathsf{C}, \mathcal{T}, M$)

Input : C: defined concept; $\mathcal{T}$: TBox; $M = \{\Sigma_1, \ldots, \Sigma_n\}$: set of *already identified* MDSs of C

Output: a potentially updated M is returned that may contain new MDSs, if M was incomplete

$\mathcal{S} \leftarrow \bigcup \ \forall\Sigma\{\Sigma \in M||\Sigma| > 1|\}$
$\mathcal{K} \leftarrow \mathsf{Sig}(\mathcal{T}) \setminus (\mathcal{S} \cup \{\mathsf{C}\})$
$\mathcal{P} \leftarrow$ POWERSET($\mathcal{S}$)
$\mathcal{P} \setminus \forall p\{p \in \mathcal{P}|\Sigma \subseteq p|\Sigma \in M\}$
for $p \in \mathcal{P}$ **do**
  $\mathcal{W} \leftarrow \mathcal{K} \cup p$
  if ISCONCEPTIMPLICITLYDEFINED($\mathsf{C}, \mathcal{T}, \mathcal{W}$) **then**
    $\Sigma_i \leftarrow$ COMPUTESINGLEMDS-D&C($\mathsf{C}, \mathcal{T}, \mathcal{W}$)
    $M \leftarrow M \cup \{\Sigma_i\}$
    $\mathcal{P} \setminus \forall p\{p \in \mathcal{P}|\Sigma_i \subseteq p\}$
  end
end
return M

4.6.1 Expanding MDSs

The algorithm for expanding MDSs has two applications: (*i*) it determines whether a given set of MDSs is complete; (*ii*) it computes new MDSs from an existing, incomplete set of MDSs. The process combines entities from existing MDSs, and from the other, non-MDS entities of the TBox signature. Any *new MDS found during the process will overlap* with existing MDSs. Figure 4.7 illustrates the expansion process that is formalised in Algorithm 12.

Walkthrough. The prerequisite of the process is that $|M| \geq 1$, i.e. there is at least

one previously computed MDS of C. First, all existing MDSs are merged into the set $\mathcal{S}$ (line 1); however, MDSs that contain only one entity are excluded from $\mathcal{S}$, as any DS based on such a signature is clearly non-minimal. Next the entity set $\mathcal{K}$ is initialised, this contains those entities of the ontology that do not appear in any existing MDSs (line 2). The initialisation concludes by computing $\mathcal{P}$, the power set of $\mathcal{S}$ (line 3), and filtering out any subset $p \in \mathcal{P}$ that contain any existing MDS (line 4).

The process then begins by exhaustively examining each subset $p \in \mathcal{P}$. From every p, a candidate definition signature $\mathcal{W}$ is generated by merging p with $\mathcal{K}$ (line 6). $\mathcal{W}$ is then tested to check whether it implicitly defines C. If so, then $\mathcal{W}$ contains at least one new MDS, thus a new MDS Σ_i is extracted and stored in M (line 8-9). Again $\mathcal{P}$ is filtered by removing any candidate signature that contains Σ_i. At the end of the process M is returned, which either contains new MDSs, thus the input was incomplete, or is the same size as it was at input, i.e. the initial M was complete. If M contains new MDSs, it indicates that there may be more undiscovered MDSs, based on the lastly identified MDSs. In this case the algorithm should be run again using the updated set of MDSs (the returned M) in order to determine whether the updated M is complete.

Correctness. Each candidate signature that implicitly defines C, is reduced to be a minimal DS by using the ComputeSingleMDS-D&C, which is proven to generate a single correct and minimal DS.

Termination and Complexity. The process exhaustively tests certain subsets of a power set of $\mathcal{S}$ (the union of all MDSs of a given entity), thus it makes an exponential number of calls to the oracle in the number of axioms in the ontology where each oracle call may itself take exponential time in the number of axioms in the ontology. The algorithm terminates when all cases have been examined. This approach employs the same optimisation as Algorithm 5, as the search space is effectively pruned by excluding every superset of each computed MDS, however the decrease in complexity is predicated on the number and the size of the MDSs.

4.6.2 Search and Expand

Algorithm 13 identifies *all MDSs* of a given concept by employing a *search then expand* strategy; Figure 4.8 illustrates, and Algorithm formalises 13 the process. The *search* phase obtains a set of mutually disjoint MDSs[1]; this set is then completed during the *expansion* phase.

Walkthrough. First the algorithm initialises $\mathcal{S}$ with the TBox signature excluding the defined concept C (line 1). Then the set of mutually disjoint MDSs of C is computed, and stored in M (line 2). Next, the algorithm enters a loop, which only terminates when

[1] Unless the given concept has multiple, overlapping explicit definitions; in this case the MDSs returned by the sub-routine overlap.

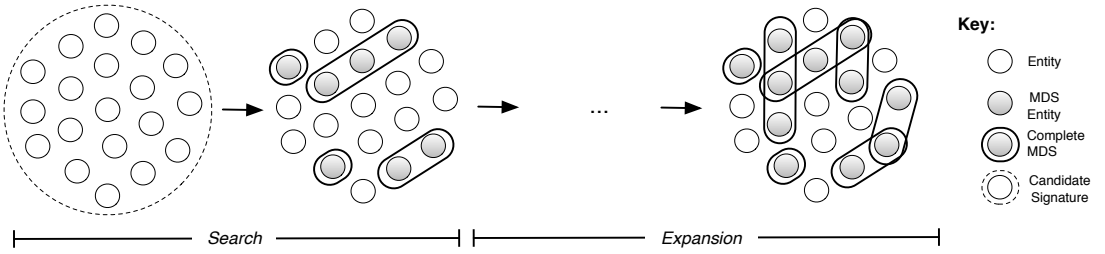


FIGURE 4.8: Depiction of a *Search and Expand* MDS computation strategy.

Algorithm 13: COMPUTEALLMDSS-S&E($\mathsf{C}, \mathcal{T}$)

Input : C: defined concept; $\mathcal{T}$: TBox
Output: $M = \{\Sigma_1, \ldots, \Sigma_n\}$: the complete set of MDSs of C

```
1  S ← Sig(T) \ {C}
2  M ← ComputeDisjointMDSs2(C, T, S)
3  while True do
4  |   M' ← ExpandMDSs(C, T, M)
5  |   if M = M' then
6  |   |   return M
7  |   end
8  |   else
9  |   |   M ← M'
10 |   end
11 end
```

all MDSs of C have been found, i.e. when M is complete (line 3-11). At each iteration M is tested for completeness using the EXPANDMDSS subroutine (line 4). If the returned M' is equivalent to the input, then M is complete, thus the process terminates (line 6). Otherwise M' contains newly found MDSs; these are then stored in M, and the process is repeated (line 9).

Completeness and Termination. The completeness of this algorithm is ensured by (i) the EXPANDMDSS algorithm, which has been shown to correctly determine the completeness of a given MDS set; (ii) the loop structure, which only terminates when there are no more unidentified MDSs. Furthermore, the process always halts after a finite number of iterations, as the maximum number of MDSs of a given entity is finite and the loop is not repeated unless a new MDS has been found during the previous iteration.

Complexity. The number of calls to the oracle is determined by two factors: (i) The number of MDSs of a given concept as the process only terminates when all MDSs are found; although, at the worst-case, this number could be exponential in the signature of the ontology, evaluation suggests that this does not typically occur in real-world ontologies.

$$\bigcup_{i=1}^{|M|} \forall \Sigma_i \{\Sigma_i \in M | 1 < |\Sigma_i|\} \quad \subseteq \quad \mathsf{Sig}(\mathcal{T} \setminus \{\mathsf{C}\}) \tag{4.1}$$

(*ii*) The expansion phase involves exhaustively processing a *power set of the union* of all priori computed MDS entities; however the cardinality of this union is significantly lower compared to the brute-force approach (Algorithm 5) which is the only other strategy presented in this thesis that finds the complete set of MDSs. The brute force approach also exhaustively tests each member of the *power set of the TBox signature* ($\mathsf{Sig}(\mathcal{T}\backslash\{\mathsf{C}\})$), which is typically much larger than the power set of the union (4.1). To summarise, the process makes exponential number of calls to the oracle in the number of axioms in the ontology where each oracle call may itself take exponential time in the number of axioms in the ontology.

4.6.3 Expansion by Rewriting

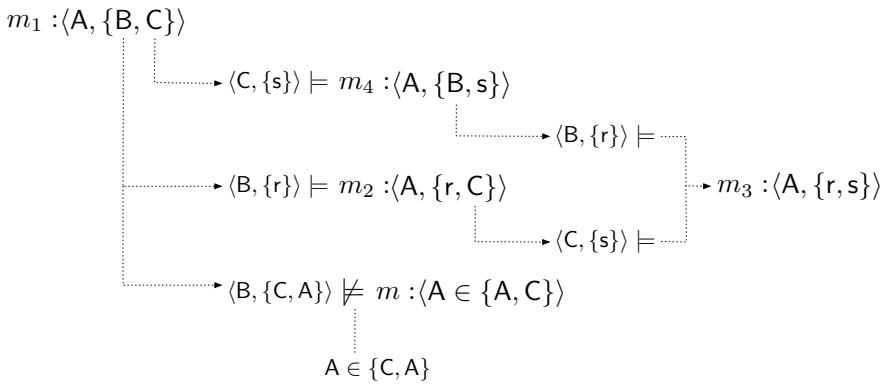


FIGURE 4.9: Expanding an existing MDS (m_1) by replacing its defined entities with their corresponding MDSs; the process potentially yields new MDSs ($m_2 - m_4$).

Algorithm 14, depicted by Figure 4.9, may expand a (non-empty) set of potentially incomplete MDSs. The process identifies new minimal DSs by replacing defined entities with all of their available MDSs, until no new MDSs are found. For example given that a concept A is implicitly definable with the signature $\{\mathsf{B}, \mathsf{C}\}$, and B is definable with $\{\mathsf{r}\}$, B can be can be replaced in the MDS of A, producing the new DS $\{\mathsf{r}, \mathsf{C}\}$. Unlike any other MDS computation algorithm, this approach finds MDSs of a *set of defined entities*, instead of MDSs for a single entities.

Walkthrough. Let $\mathbf{M}$ be a set of tuples such that $m_i : \langle \mathsf{e}_i, \Sigma\rangle$ where e_i denotes a defined entity and Σ denotes the MDS of e_i; moreover the function $\mathsf{e}(m_i)$ returns e_i, and $\Sigma(m_i)$ returns Σ. For each existing MDS $m_i \in \mathbf{M}$ the process examines each constituent entities of the signature $\Sigma(m_i)$ (line 2-3); if a MDS entity $\mathsf{e}_j \in \Sigma(m_i)$ is defined (line 4) then it can potentially be replaced in the MDS of e_i in order to obtain another, also correct MDS that describes $\mathsf{e}(m_i)$. For each defined MDS entity e_j, the GETALLMCDS subroutine locates all corresponding MDSs and stores it in a set $\mathcal{W}$ (line 5). Then the process attempts to create a set of new MDSs, by replacing e_j in m_i with each of its MDS (i.e. $m_j \in \mathcal{W}$). The process excludes any m_j which contains the defined entity (line 7). Next the process generates a new DS m_i' (line 8) and assesses its minimality (line 9).

Algorithm 14: RewriteDefinedEntities($\mathcal{C}, \mathcal{T}, \mathbf{M}$)

Input : $\mathcal{T}$: TBox; $\mathsf{E}_{Def} = \{\mathsf{e}_1, \ldots, \mathsf{e}_m\}$ defined entities, where $\mathsf{E}_{Def} \subseteq \mathsf{Sig}(\mathcal{T})$;
$\mathbf{M} = \{m_1 : \langle \mathsf{e}_i, \Sigma \rangle, \ldots, m_n\}$: MDSs of defined entities
Output: $\mathbf{M}$: a *potentially expanded* set of MDSs

```
while True do
    for each m_i ∈ M do
        for each e_j ∈ Σ(m_i) do
            if e_j ∈ E_Def then
                W ← GetAllMDS(e_j, M)
                for each m_j ∈ W do
                    if e(m_i) ∉ Σ(m_j) then
                        m'_i ← RewriteEntityWithMDS(e_j, Σ(m_j), m_i)
                        m'_i ← CheckDSMinimality(e(m_i), T, m'_i)
                        M_updated ← M ∪ {m'_i}
                    end
                end
            end
        end
    end
    if |M_updated| = |M| then
        return M
    end
    else
        M ← M_updated
    end
end
```

The resulting minimal DS m'_i is then added to $\mathbf{M}_{updated}$ (which is a set that contains no duplicate MDSs). The process is repeated for all MDSs; if $\mathbf{M}_{updated}$ is the same as $\mathbf{M}$ then no new MDSs have been found during the last iteration, thus the process terminates.

Correctness. All MDSs generated during the process are valid. Although any DS created by replacing a constituent defined entity with its MDS is a correct DS (by induction), it is not guaranteed to be minimal. For instance, let us consider the following example where $\mathbf{M} = \{m_1 : \langle \mathsf{A}, \{\mathsf{B}, \mathsf{C}\} \rangle, m_2 : \langle \mathsf{B}, \{\mathsf{r}\} \rangle, m_3 : \langle \mathsf{C}, \{\mathsf{r}\} \rangle\}$. The MDS m_1 which defines A with the signature $\{\mathsf{B}, \mathsf{C}\}$, contains a defined entity B. Replacing B in $\Sigma(m_1)$ produces the DS $m'_1 : \langle \mathsf{A}, \{\mathsf{r}, \mathsf{C}\} \rangle$, which is not minimal, as due to m_3 C is redundant in $\Sigma(m'_1)$. Thus signature minimality needs to be checked; this is ensured by the subroutine CheckDSMinimality.

Termination and Completness. The rewriting process is performed within a loop, which terminates only when there are no new MDSs found during the last iteration; ensuring the completeness of the algorithm. Termination is achieved by comparing the number of MDSs before and after the process. Once there are no more new MDSs generated, the algorithm terminates. The process is *not guaranteed* to find new MDSs.

Complexity. As the maximum number of possible MDSs defined entities is exponential in the size of the ontology signature, the worst case complexity of the process is also exponential in the size of the ontology, both in terms of the number of oracle calls made by the algorithm, and the number of iteration of the main loop. However, the number of implicit definability checks, which are performed to ensure DS minimality, is generally low due to the typically small number of redundant entities in the examined DSs.

4.7 The Impact of Modularisation

As discussed in Chapter 2.2.4, a module $\mathcal{M}$ is a independent subset of a TBox $\mathcal{T}$ such that $\mathcal{M}$ implies the same concept inclusions for its own subject matter (a signature) as $\mathcal{T}$. Therefore, an $\mathcal{S}$-module of a given concept C, where $\mathcal{S} = \{\mathsf{C}\}$, by definition preserves all entailments, of C with respect to $\mathcal{T}$:

$$\mathcal{T} \models \mathsf{C} \equiv \mathsf{D} \;\; \Leftrightarrow \;\; \mathcal{M} \models \mathsf{C} \equiv \mathsf{D} \tag{4.2}$$

where D is a potentially complex concept which implicitly defines C. Therefore, at most the number of unique MDSs of a given defined concept (or role) is exponential in the size of an $\mathcal{S}$-module of the defined entity (in the worst case a module is equivalent to the TBox). As modules can be considerably smaller compared to the original ontology, and can be efficiently computed, *modularisation is an effective mechanism for reducing the complexity of definability computation.*

4.7.1 Conjecture.

Let $\mathcal{T}$ be an $\mathcal{ALC}$ TBox and $\mathcal{T} \models \mathsf{A} \equiv \mathsf{C}$ for an $\mathcal{ALC}$ concept C not containing A such that there is no $\mathcal{ALC}$ concept C' not containing A whose signature is properly contained in the signature of C with $\mathcal{T} \models \mathsf{A} \equiv \mathsf{C}'$. Then $\mathcal{M} \models \mathsf{A} \equiv \mathsf{C}$ for the $\top\bot^*$-module $\mathcal{M}$ of $\mathcal{T}$ with seed signature $\{\mathsf{A}\}$ and, moreover, the signature of C is contained in the signature of $\mathcal{M}$.

Proof. It suffices to prove this for the $\top$-module $\mathcal{M}^{\top}$ and the $\bot$-module $\mathcal{M}^{\bot}$ of $\mathcal{T}$ with seed signature $\{\mathsf{A}\}$. We show this for $\mathcal{M}^{\bot}$. The proof for $\mathcal{M}^{\top}$ is similar and omitted. Thus, let $\mathcal{T} \models \mathsf{A} \equiv \mathsf{C}$. Replace any concept name in the signature of C that is not in the signature of $\mathcal{M}^{\bot}$ by $\bot$, replace any $\exists \mathsf{r}.\mathsf{F}$ with r in the signature of C and not in the signature of $\mathcal{M}^{\bot}$ by $\bot$ and replace any $\forall \mathsf{r}.\mathsf{F}$ with r in the signature of C and not in the signature of $\mathcal{M}^{\bot}$ by $\top$. Denote the resulting concept by C'.

Let $\mathcal{I}$ be any model of $\mathcal{M}^{\bot}$. Let $\mathcal{I}'$ be the model obtained from $\mathcal{I}$ by interpreting every symbol $\neq \mathsf{A}$ and not in the signature of $\mathcal{M}^{\bot}$ as the empty set. Then $\mathcal{I}' \models \mathcal{T}$ by the definition of $\bot$-modules. Thus, $\mathcal{I}' \models \mathsf{A} \equiv \mathsf{C}$ and so $\mathcal{I} \models \mathsf{A} \equiv \mathsf{C}'$. It follows that $\mathcal{M}^{\bot} \models \mathsf{A} \equiv \mathsf{C}'$ and so $\mathcal{T} \models \mathsf{A} \equiv \mathsf{C}'$. Thus $\mathsf{C} = \mathsf{C}'$ by the minimality condition on the signature of C. Hence $\mathcal{M}^{\bot} \models \mathsf{A} \equiv \mathsf{C}'$.

4.7.2 Determining Definability.

Determining whether a concept has an *explicit defintion* (Algorithm 1) of a is achieved by searching through the set of axioms $\mathsf{Ax}(\mathcal{T})$, for an explicit concept definition of C. Thus the search space is reducible to $\mathsf{Ax}(\mathcal{M})$, providing the linear decrease given by the number $n = |\mathsf{Ax}(\mathcal{T} \setminus \mathcal{M})|$.

Determining *implicit definability* (Algorithm 2) of a concept name C with respect to a signature Σ under a TBox $\mathcal{T}$ is implemented as a two step process: first $\mathcal{T}'$, a copy of the TBox is created and every occurrence of every entity of $\mathsf{Sig}(\mathcal{T}) \setminus \Sigma$ is renamed in $\mathcal{T}'$. The second step is delegated to a reasoner that computes whether the entailment (i.e. the implicit definability) $\mathcal{T} \cup \mathcal{T}' \models \mathsf{C} \equiv \mathsf{C}'$ holds. Clearly, both steps benefit from modularisation: instead of the TBox, the module is duplicated, resulting in $\mathcal{M}'$, where the number of entities required to be renamed is reduced by $n = |\mathsf{Sig}(\mathcal{T} \setminus (\mathcal{M} \setminus \Sigma))|$. The search space, i.e the set of axioms used by a reasoner to compute the entailment is decreased to $\mathcal{M} \cup \mathcal{M}'$, hence the difference in terms of the number of axioms equals to $2 \cdot |\mathsf{Ax}(\mathcal{T} \setminus \mathcal{M})|$.

4.7.3 Justifying definability.

Algorithm 4 determines implicit definability by computing justifications (sets of axioms explaining definability) for the entailment $\mathcal{T} \cup \mathcal{T}' \models \mathsf{C} \equiv \mathsf{C}'$. Modularisation only affects the initialisation phase, which is identical to the first part of the implicit definability check procedure. Computing justifications does not benefit from modularisation, because modularity is already used by the algorithm to produce justifications, that computes entailments from modules instead of the entire ontology (Section 3.2.1 in [73]).

4.7.4 MDS Search.

Base MDS search algorithms (Algorithm 6, 7) produce a *single MDS*, by processing definition signatures into minimal DSs. This is achieved by iteratively pruning the input set: these algorithms either examine the input one entity at the time, or as sets of entities, in order to determine the required and the redundant members of a given DS. In this case, modularisation is carried out prior to invoking the search process. Without modularisation, the input signature is at most the set $\mathsf{Sig}(\mathcal{T}) \setminus \{\mathsf{C}\}$, with modularisation this is reduced to $\mathsf{Sig}(\mathcal{M}) \setminus \{\mathsf{C}\}$. Clearly every algorithm, e.g. Algorithm 10 (which computes set of mutually disjoint MDSs), that use base MDS search algorithms as subroutines also benefit from modularisation.

4.7.5 MDS Expansion.

Algorithm 4.6.1 finds new MDSs by combining entities of existing MDSs with the rest of the ontology signature (i.e. non-MDS entities). Modularisation is applied when a candidate signature is generated; each candidate signature is a union of two parts: one comes from existing MDSs, the other part is the rest of the ontology signature

(non-MDS entities) that can be reduced to the module signature, excluding any MDS members ($\mathsf{Sig}(\mathcal{M}) \setminus \bigcup_{i=1}^{|M|} \Sigma_i$). The process benefits from modularisation when the candidate signature is tested for definability, and when an MDS is extracted from a candidate signature.

4.8 Summary and Conclusions

- Finding a single MDS is potentially a computationally expensive process, as both the number of the possible unique MDSs of a given defined entity, and the number of candidate signatures required to be explored in order to confirm completeness of MDSs, is exponential in the size of the ontology. Furthermore, finding all MDSs for a given defined concept or role, using a *naive approach* requires exponential number of calls to the oracle in the number of axioms in the ontology where each oracle call may itself take exponential time in the number of axioms in the ontology.

- This chapter has also presented a *pragmatic approach* to computing the complete set of MDSs for a given ontology. The approach can identify a subset of all possible MDSs by using a polynomial number of calls to the oracle in the number of axioms in the ontology where each oracle call may itself take exponential time in the number of axioms in the ontology.

- This chapter has explored the use of *modularisation*, which was shown to be an effective mechanism for reducing the complexity of definability computation.

- Because little is known about the actual cost of the implicit definability check in real world ontologies, and expansion search algorithms have an exponential worst case complexity in terms of the number of oracle calls made by the algorithms where each oracle call may itself take exponential time in the number of axioms in the ontology, there is a need for empirical data that characterises the behaviour of the presented algorithms.

Chapter 5

Minimal Definition Signature Evaluation

This chapter presents an empirical investigation of the approaches to find Minimal Definition Signatures, which was performed over a wide range of OWL ontologies, in order to assess the prevalence of definability scenarios, and to evaluate the behaviour of the proposed algorithms for computing definability. First, Section 5.1 describes the large and diverse ontology corpus that has facilitated the empirical evaluation of definability computation and the approaches that exploit definability (i.e. make use of precomputed minimal definition signatures). Section 5.2 presents several evaluations that were carried out as part of the empirical investigation. The first evaluation focuses on determining which ontology characteristics (if any) are affected by definability. The goal of the second is to evaluate the performance and practicality of the proposed algorithms, and to measure the impact of modularisation in definability computation. Section 5.3 summarises and concludes the chapter.

5.1 The Evaluation Corpus

This section describes the selection, curation and the general properties of the evaluation corpus utilised within the various experiments presented throughout the thesis.

Requirements. The evaluation corpus was assembled from several OWL datasets. The *OWL ontology format* was chosen for a number of reasons. As previously described, OWL is the official W3C recommendation knowledge representation language for authoring ontologies on the Web. OWL is popular both in academia and in the commercial world. As a consequence there is a plethora of *accessible and freely obtainable OWL ontologies*, in the form of individual files and curated collections. Furthermore, OWL has *excellent tool support*; this was important for the implementation of the experiment framework, which required: (*i*) an API for creating, manipulating and processing ontologies, and in particular performing module extraction; ontology reasoner(s); and (*ii*) the API for computing justifications, which is exclusive to OWL.

The creation of a corpus is necessary to facilitate a number of *different type of evalations*. The first set of experiments, presented in this chapter, investigates implicit definability across numerous ontologies in order to highlight those properties that may effect the prevalence and the extent of implicit definability. Moreover, these experiments are meant to identify a set semantically rich ontologies (i.e. documents with high ratio of implicitly definable entities and number of MDSs) to support experiments exploiting the notion of implicit definability. For this purpose, dataset selection was dictated by *diversity*; i.e. the dataset must contain a wide range of ontologies of different *size* (with respect to signature $|\mathsf{Sig}(\mathcal{O})|$, as well as axiomatisation $|\mathsf{Ax}(\mathcal{O})|$), and *language expressivity* (ranging from simple to very expressive). In addition, it was important that the corpus consisted of *real world* ontologies depicting a *variety of domains*, with a broad range of *application areas*[1], and originating from a large number of independent *sources* (domain experts, ontology engineers, application developers, etc.) that may apply different modelling styles. This was desirable if the generality of the approach was to be assessed. Lastly, it was also crucial to have a sufficient number of ontologies in order to uncover any patterns or anomalies that may effect the derived conclusions. The definability status of entities and the MDSs computed during these experiments would also be used to support the empirical evaluation of approaches that exploit definability, presented in the later chapters.

As previously discussed (Section 3.6), implicit definability check works in languages that do not accept Beth definability, thus whether the Beth definability property holds for a given ontology language is not crucial for the practical applications of this work. Therefore the experiment corpus was not filtered by checking which ontology language accept Beth definability.

5.1.1 Corpus Selection and Curation

Matentzoglu et al. presented an overview of the OWL ontology landscape and provided a comprehensive picture about several important ontology collections and repositories [95, 96]. Furthermore, the authors described good practices and pitfalls of selecting, curating, analysing and comparing datasets for empirical evaluation. Thus, this methodology was followed in creating and curating the evaluation corpus described here. The evaluation corpus was assembled from six different collections, which included two large (i.e. hundreds of documents) and one very large (i.e. thousands of documents) datasets that supported the investigation of definability on a diverse set of ontologies:

- **BioPortal corpus.** This hand crafted, community-based repository is hosted by the National Center for Biomedical Ontology (NCBO) [105]. According to

[1]The purpose for which a particular ontology is used impacts the ontology engineers' modelling choices, such as DL expressivity etc. For example, a highly expressive language can be detrimental to the effectiveness of different reasoning services. Therefore, one would not use such a language when the aim is to perform large scale or frequent inferences. On the other hand, when the goal of the conceptualisation is to model the most accurate representation of some domain of interest, an expressive language is usually a better choice.

the organisation's website[2], it is the 'world's most comprehensive repository of biomedical ontologies'. These real world ontologies vary greatly in size and expressivity, which makes it a popular corpora for tool development and academic empirical evaluation. This corpora was also used for evaluating the computation of justification-based explanations (Chapter 5, [73]) which, in the context of this research, is used for validating MDSs.

- **TONES corpus.** This hand-curated ontology repository was developed to provide a comprehensive test set for OWL application development and empirical studies[3]. At the time of access, it contained 219 OWL and OBO ontologies, varing greatly in size and expressivity.

- **WebCrawl corpus.** This corpus was created by Matentzoglu et al. [96], and was obtained by crawling the web and collecting OWL ontologies. The dataset was curated by applying various filtering heuristics (file and domain based manual cleaning procedures, repairing some minor syntactic errors such as missing entity declarations, etc.), resulting in a very large (in comparison to the other listed corpora) set of non-trivial OWL ontologies containing 4327 documents. The main purpose of this corpus is to allow sampling based empirical evaluation, yielding more representative results than when cherry-picking individual documents, or somewhat arbitrary selecting certain data sets etc.

In addition, three small datasets were selected to facilitate the evaluation of the approaches (presented in Chapter 7 and 8), that exploit implicit definability to support semantic interoperability between heterogenous ontologies, i.e. through ontology matching and alignment negotiation. These curated datasets are maintained by the OEAI, and support the alignment evaluation challenge in assessing several different aspects of ontology matching systems and the resulting alignments, over different campaign *tracks*. Each track consists of different datasets, designed to evaluate certain ontology matching features (e.g. instance matching, large ontology matching, interactive matching, etc.) [46]. Furthermore, each track contains reference alignments, as well as the alignments produced by the competing approaches. The following tracks were selected:

- **Anatomy track.** This track contains two large ontologies; one describes the human anatomy, whereas the other represents the anatomy of mice.

- **Large Biomedical Ontologies.** This dataset[4] consists of 6 large, semantically rich ontologies, that are extracts (overlapping fragments) of three well known biomedical ontologies: Foundational Model of Anatomy (FMA) [114], SNOMED CT [36], and the National Cancer Institute Thesaurus (NCI) [104].

[2] `http://bioportal.bioontology.org/`
[3] `http://rpc295.cs.man.ac.uk:8080/repository/`
[4] `http://oaei.ontologymatching.org/2014/largebio/index.html`

	Anatomy	LargeBio	Conference	TONES	BioPortal	WebCrawl	**Total**
Obtained	2	6	15	229	251	3903	**4406**
Processed	2	6	15	178	204	3005	**3410**
	100.00%	100.00%	100.00%	77.72%	81.27%	76.99%	**77.39%**

Table 5.1: Number of documents in the six collections before curation, and the percentage of the datasets that were processed (i.e. curated, and the implicit definability check was completed).

- **Conference track.** This track[5] containing a collection of 16 well-defined, small ontologies that model the conference organisation domain. This particular track is used by several ontology alignment evaluation and alignment negotiation approaches that were mentioned in Chapter 7 and 8, respectively.

Curation Process. Horridge et al. published a curated snapshot of the *BioPortal* repository, and the *WebCrawl* corpus [75]. These ere obtained[6] and used as is. The rest of the evaluation corpus, including the three *OAEI tracks* and the *TONES repository* snapshot, was curated as follows: each document was parsed using the OWL API, and any file that could not be parsed or was incomplete (i.e. contained missing imports[7]) was discarded. In addition, each document was tested for *consistency*, because as previously described, inconsistency in an ontology leads to false implicit definability results. Thus, all inconsistent documents were discarded. The consistency check was performed using either the Hermit or the Pellet ontology reasoner; Pellet supported those documents that would not be processed by Hermit due to transitive property declarations. As large collections may overlap, i.e. contain the same or similar ontologies [95, 96]; *identical duplicate* documents were removed. However, *versions* (different releases of the same ontology) and *variants* (the 'same' ontology with different DL expressivity, for example light or full) were kept. Two ontologies are considered *duplicates* if their terminological parts match (signature, TBox and RBox); the ABox was not considered when determining similarity. Furthermore, several large or very expressive ontologies could not be processed (meaning that the implicit definability checking algorithm could not complete its run) due to either *reasoner time-out error* (with 10 minute time-out setting), or *memory overflow* of the Java Virtual Machine (8GB allocated for the JVM). These documents were also excluded from the final, curated versions of the corpus. Table 5.1 lists the resulting ontologies in each of the six sub-corpora prior to curation, and the ratio of the processable ontologies in a sub-corpora, after curating and conducting the implicit definability check. Overall 77.39% of all documents, i.e. 3410 ontologies were processed.

[5] `http://oaei.ontologymatching.org/2014/conference/index.html`

[6] `http://web.stanford.edu/~horridge/publications/2014/iswc/atomic-decomposition/data/`

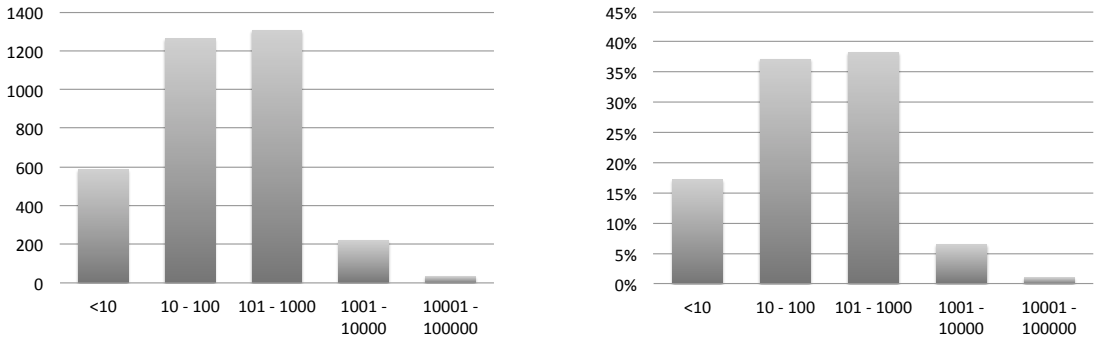
[7] OWL ontologies, much like software, may be built in a modular fashion, where reusable ontological knowledge is kept physically separate (i.e. it is an ontology on its own), but can be included to create a single document through importing; top-level or foundation ontologies that describe very general concepts are often used across many different knowledge domains. An ontology which imports others is dependent on its imports, thus any imports must be available (online or locally), otherwise the ontology cannot be loaded. To avoid scenarios where imports cannot be loaded, or to reduce loading time (especially with large documents stored online), these are often merged with the 'base' ontology.

5.1.2 Corpus Properties

The main properties of the processed corpus are detailed in this subsection, in order to demonstrate the diversity of the corpus w.r.t. several characteristics.

		Anatomy	LargeBio	Conference	TONES	BioPortal	WebCrawl
Signature	min	2755	3725	32	1	6	2
	avg	3034	18187	113	1451	2450	119
	med	3034	11809	109	168	569	53
	max	3313	51181	274	36090	45091	2744
Classes	min	2743	3696	14	0	0	1
	avg	3024	18140	54	1239	2060	61
	med	3024	11785	49	88	451	15
	max	3304	51128	140	36076	38738	2329
Object properties	min	2	0	13	0	0	0
	avg	3	36	33	33	39	20
	med	3	35	33	5	11	5
	max	3	82	58	922	1390	607
Data properties	min	0	24	0	0	0	0
	avg	0	8	12	14	9	9
	med	0	0	11	0	0	1
	max	0	24	23	708	488	674
Individuals	min	0	0	0	0	0	0
	avg	0	0	10	40	341	19
	med	0	0	0	0	0	1
	max	0	0	114	3542	22534	1014
Logical Axioms	min	4838	3828	65	0	3	0
	avg	8192	24150	265	1882	4156	215
	med	8192	15268	233	187	839	69
	max	11545	71042	739	42656	77700	4556

Table 5.2: Entity usage (minimum, average, median, maximum) in the six corpora.



(a) absolute distribution (a) relative distribution

Figure 5.1: Distribution of ontology sizes in the corpus, sorted by the number of logical axioms.

Entity Usage. Table 5.2 shows the signature size, the entity usage (classes, properties, individuals), and the number of logical axioms in the six corpora, see caption (5.2). In addition, Figure 5.1 shows the distribution of ontology sizes sorted into six size-bins according to the number of logical axioms. In terms of the logical axiom count, the main corpus (i.e. the union of the six collections) is sufficiently diverse in size: the majority falls into the *very small* (less than 10 axioms) and *small* (10 to 100 axioms) size-bins

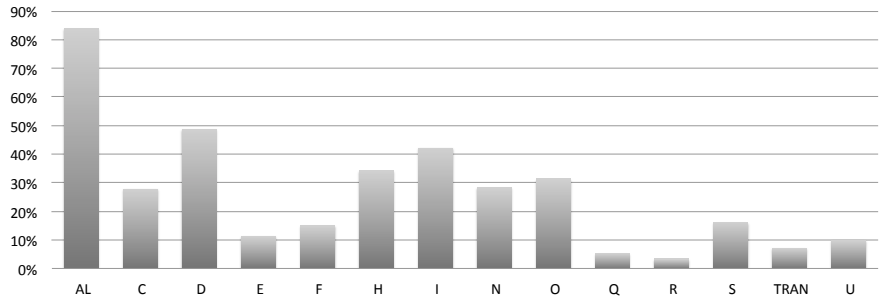


FIGURE 5.2: Frequency of OWL constructor usage (where the x-axis denotes the constructors, and the y-axis denotes the frequency of ontologies using the particular constructor).

	ANATOMY	LARGEBIO	CONFERENCE	TONES	BIOPORTAL	WEBCRAWL	TOTAL
Full	0.0%	0.0%	0.0%	16.7%	41.2%		**42.2%**
DL	100.0%	100.0%	100.0%	83.3%	58.8%		**57.8%**
EL only	0.0%	33.3%	0.0%	10.6%	1.9%		**1.9%**
EL total	0.0%	33.3%	0.0%	31.1%	6.9%		**7.6%**
QL only	0.0%	0.0%	0.0%	0.0%	1.0%		**1.1%**
QL total	0.0%	0.0%	0.0%	20.0%	5.2%		**5.6%**
RL only	0.0%	0.0%	6.7%	2.2%	5.0%		**7.5%**
RL total	0.0%	0.0%	6.7%	13.9%	9.3%		**13.4%**
DL only	0.0%	66.7%	93.3%	49.4%	45.1%		**43.8%**

TABLE 5.3: OWL 2 profile distribution in the collections.

(with 17% and 37%, respectively); *medium* size (101 to 1000 axioms) ontologies constitute 38% of the dataset; about 6% is *large* (1001 to 10000 axioms), and 1% falls into the *very large* (over 10000 axioms) size category, making up the remainder of the corpora.

Expressivity. We looked at the expressivity of the used ontologies. Figure 5.2 presents the frequency of OWL constructors usage in the whole evaluation corpus. The different DLs fall within the various OWL 2 profiles (described in Section 2.3). In addition to the three named profiles, the 'DL' category includes those ontologies that do not fall into any of the named profiles, but that represent a valid syntactic subset of the OWL 2 language. The 'Full' category denotes those OWL documents that cannot be classified as 'DL', but were still parsable by the OWL API and processable by reasoners (this could occur when, for instance, entity declarations are missing from the document). Table 5.3 shows the named profiles in the corpus, for each sub-collection, as well as the whole corpus. As an ontology can simultaneously fall within more than one named profile, the table provides an additional category which denotes those ontologies that adhere exclusively to a single profile. As the evaluation corpus contained ontologies with over 250 different constructor combinations (these omitted for the sake of readability), and provides examples of all OWL 2 profiles, it is sufficiently diverse in terms of DL expressivity, for the purpose of this evaluation.

Axiom Usage. OWL categorises the axioms into different *axiom types*; for example a

Axiom type	Anatomy	LargeBio	Conference	TONES	BioPortal	WebCrawl
EquivalentClasses	0.0%	34.6%	66.7%	39.3%	45.6%	38.1%
SubClassOf	100.0%	100.0%	100.0%	97.0%	87.5%	76.2%
DisjointClasses	50.0%	88.5%	66.7%	53.5%	42.7%	35.1%
DisjointUnion	0.0%	0.0%	0.0%	0.0%	0.5%	0.0%
ClassAssertion	100.0%	23.1%	0.0%	30.8%	22.4%	51.2%
EquivalentObjectProperty	0.0%	0.0%	0.0%	5.5%	2.2%	2.2%
SubObjectPropertyOf	0.0%	36.5%	50.0%	50.6%	34.0%	33.2%
DisjointObjectProperty	0.0%	0.0%	0.0%	3.2%	0.4%	0.2%
ObjectPropertyDomain	0.0%	100.0%	33.3%	51.7%	42.3%	54.6%
ObjectPropertyRange	0.0%	100.0%	33.3%	49.0%	43.0%	56.3%
ObjectPropertyAssertion	0.0%	0.0%	0.0%	11.0%	8.7%	16.5%
NegativeObjectPropertyAssertion	0.0%	0.0%	0.0%	0.0%	0.0%	0.9%
InverseObjectProperties	0.0%	0.0%	0.0%	45.4%	28.3%	38.3%
TransitiveObjectProperty	50.0%	32.7%	0.0%	46.7%	27.4%	21.5%
SymmetricObjectProperty	0.0%	7.7%	0.0%	18.5%	12.9%	11.4%
AsymmetricObjectProperty	0.0%	0.0%	0.0%	1.6%	1.0%	0.7%
FunctionalObjectProperty	0.0%	63.5%	0.0%	32.4%	23.7%	22.8%
InverseObjectProperty	0.0%	51.9%	0.0%	7.9%	8.4%	9.2%
IrreflexiveObjectProperty	0.0%	0.0%	0.0%	3.8%	0.4%	0.7%
SubPropertyChainOf	0.0%	0.0%	33.3%	9.6%	2.2%	1.6%
EquivalentDataProperty	0.0%	0.0%	0.0%	1.1%	0.5%	1.7%
SubDataPropertyOf	0.0%	3.8%	0.0%	12.0%	1.4%	10.4%
DisjointDataProperty	0.0%	0.0%	0.0%	0.0%	0.0%	0.1%
DataPropertyDomain	0.0%	71.2%	33.3%	26.5%	21.3%	37.5%
DataPropertyRange	0.0%	71.2%	33.3%	28.0%	25.3%	38.7%
FunctionalDataProperty	0.0%	59.6%	33.3%	19.9%	16.8%	19.9%
DataPropertyAssertion	0.0%	0.0%	0.0%	11.9%	4.0%	13.5%
SameIndividual	0.0%	0.0%	0.0%	0.6%	1.1%	17.4%
DifferentIndividuals	0.0%	3.8%	0.0%	11.8%	3.2%	6.0%

Table 5.4: Axiom type usage in the six collections, as a proportion of ontologies that use an axiom type.

concept synonym statement such as $\mathsf{C} \equiv \mathsf{D}$ is formalised by the *EquivalentClasses* axiom. Table 5.4 shows the axiom type usage in each of the subsets of the evaluation corpus.

5.2 Empirical Evaluation

This section empirically investigates the occurrence of definability in existing ontologies, and the impact it has in supporting semantic interoperability. In addition, it analyses the behaviour of the proposed algorithms to compute the definability of ontological expressions. The underlying assumption made here is that the definability status (*undefined*, or *defined*: *explicitly* and/or *implicitly*) of ontology signature entities, and the number of MDSs of defined entities provide a measure of the usability of an ontology in semantic interoperability[8]. Thus, in order to gain insight on whether, in practice, the use of definition signatures would contribute to ontology alignment in particular, and ontology engineering in general, the investigation assessed the *prevalence and the extent of definability* over a wide range of OWL ontologies. Furthermore, the investigation: (*i*) studied the behaviour of the proposed approximations to compute MDSs in terms of run time taken for each of the stages necessary to compute the MDSs; (*ii*) compared

[8]For example, given two versions of an ontology, the one with more defined entities or higher MDS to entity ratio is more valuable, as it may permit the expression of more entities with alignments that are typically incomplete [47].

various MDS computation approaches; and (*iii*) analysed the impact of modularisation to definability computation.

The first experiment (Section 5.2.1) divides ontologies into two categories: *(i) ontologies that contain entities which has MDSs* (denoted as *HasMDS*), i.e. there is at least one either explicitly defined or implicitly definable entity in the given ontology; *(ii) ontologies where none of the entities has MDSs* (denoted as *HasNoMDS*), i.e. none of the entities of the given ontology are explicitly defined or implicitly definable; and examines the definability status and type for each entity in all the ontologies of a large and diverse corpus. Furthermore, it considered several characteristics of the defined and the undefined ontologies.

An additional aim of this empirical analysis (Section 5.2.2) is to *characterise the behaviour and assess the practical applicability* of the proposed definability computation algorithms. This is achieved by measuring the processing time when computing MDSs in a corpus consisting of 'semantically rich' ontologies, i.e. that contain a large portion of either explicitly defined or implicitly definable entities and MDSs. As previously described in Section 4.1, definability computation is a three step process: first the definability status of each entity is established, next the disjoint MDSs are obtained, finally any potentially unidentified MDS is computed (i.e. the complete set of MDSs). While during the first two steps the algorithms use polynomial number of calls to the oracle in the number of axioms in the ontology where each oracle call may itself take exponential time in the number of axioms in the ontology, the third step uses exponential number of calls to the oracle. In addition, the experiment compares two MDS computation approaches, *single* and *multi-entity pruning*, that produce one MDS at the time; while the former approach uses linear number (the size of a given input signature) of oracle calls, the later performs better (logarithmic time) in the best case, but potentially takes twice as long as the former approach, in the worst case.

Modularisation (Section 4.7) can be an effective mechanism for reducing the complexity of definability computation. As a module typically contains a subset of the axioms found in the original TBox, modularisation can potentially improve the performance of the algorithms used to compute definability. Experiment 3 (Section 5.2.3) analyses the impact of modularisation on the definability computation, by comparing the size (in terms of number of axioms and signature cardinality) of TBox-es and modules, and the time taken by the implicit definability check method.

Determining role definability. As previously discussed in 3.3.5, the implementation of checking implicit role definability has been restricted to RBox axioms and signatures only consisting of role names. Therefore the empirical evaluation does not identifying cases of implicitly defined roles, where both TBox, RBox and ABox axioms are used to define concepts.

Experimental Framework. The experimental framework used to run this analysis is

implemented in Java; the OWL API is used for ontology manipulation, for interacting with the reasoners [74], and for computing modules; whilst the OWL Explanation API [73] is used to compute justifications. The framework utilizes both the HermiT [63] and Pellet [126] reasoners. Whilst HermiT performs faster with most datasets, Pellet was able to load and process some ontologies that HermiT could not (due to ontologies using datatypes that are not part of the OWL 2 datatype map and no custom datatype definition was given). All of the data and software, including computed DSs, definition patterns, and other results are available online[9].

Experimental Conduct. Due to the large size of the corpus, Experiment 1 (which assesses the prevalence of definability in 4406 ontologies) was conducted over a number of different machines of a shared cluster, with the average configuration of a minimum of 16GB RAM and a 4-core processor architecture. As a consequence of this, it is not possible to make direct comparisons w.r.t. the computation time across all documents. Experiments 2 and 3, which includes evaluating the computation time of definability computation algorithms in a sample corpus of 9 ontologies (6 small and 3 large), were carried out on the same machine with 128GB RAM and a 12-core processor architecture (2 cores were reserved for system process), running a maximum of 10 process at any time. For each process, 8GB RAM was allocated for the JVM to minimise any variability in the execution environment, and thus facilitate a comparison of the time results between smaller and larger ontologies. It is worth mentioning that the definability computation can be parallelised, as the result of establishing the definability status, or finding the MDSs of a given entity, is independent of other entities.

5.2.1 Experiment 1: Prevalence and Extent of Definability

In order to determine which ontology characteristics (if any) are affected by definability, the first experiment examines each concept in all 3410 ontologies with the aim of determining whether there is at least one either explicitly defined or implicitly definable concept in the given ontology (*HasMDS*), or none of the concepts of the given ontology are explicitly defined or implicitly definable (*HasNoMDS*). Therefore, we aim to assess the prevalence of concepts with MDSs in our corpus of commonly used ontologies. The hypothesis tested in this experiment is that *definability occurs in ontology irrespective on the ontology characteristics e.g. size, expressivity etc.*

The analysis of the entire corpus classifies 1701 ontologies (49.89%) as *HasMDS*, and the rest as *HasNoMDS*. Out of all concepts in all ontologies, 75.82% are neither explicitly defined or implicitly definable, 20.74% are explicitly defined and 3.44% are only implicitly definable.

Figure (5.3.d) shows the proportion of ontologies in the corpus, binned by the ratio of concepts that have MDSs to concepts that have no MDSs within an ontology. Figure

[9] `http://www.csc.liv.ac.uk/~dgeleta/ontodfn.html` and `https://bitbucket.org/dgeleta/owl-definability`

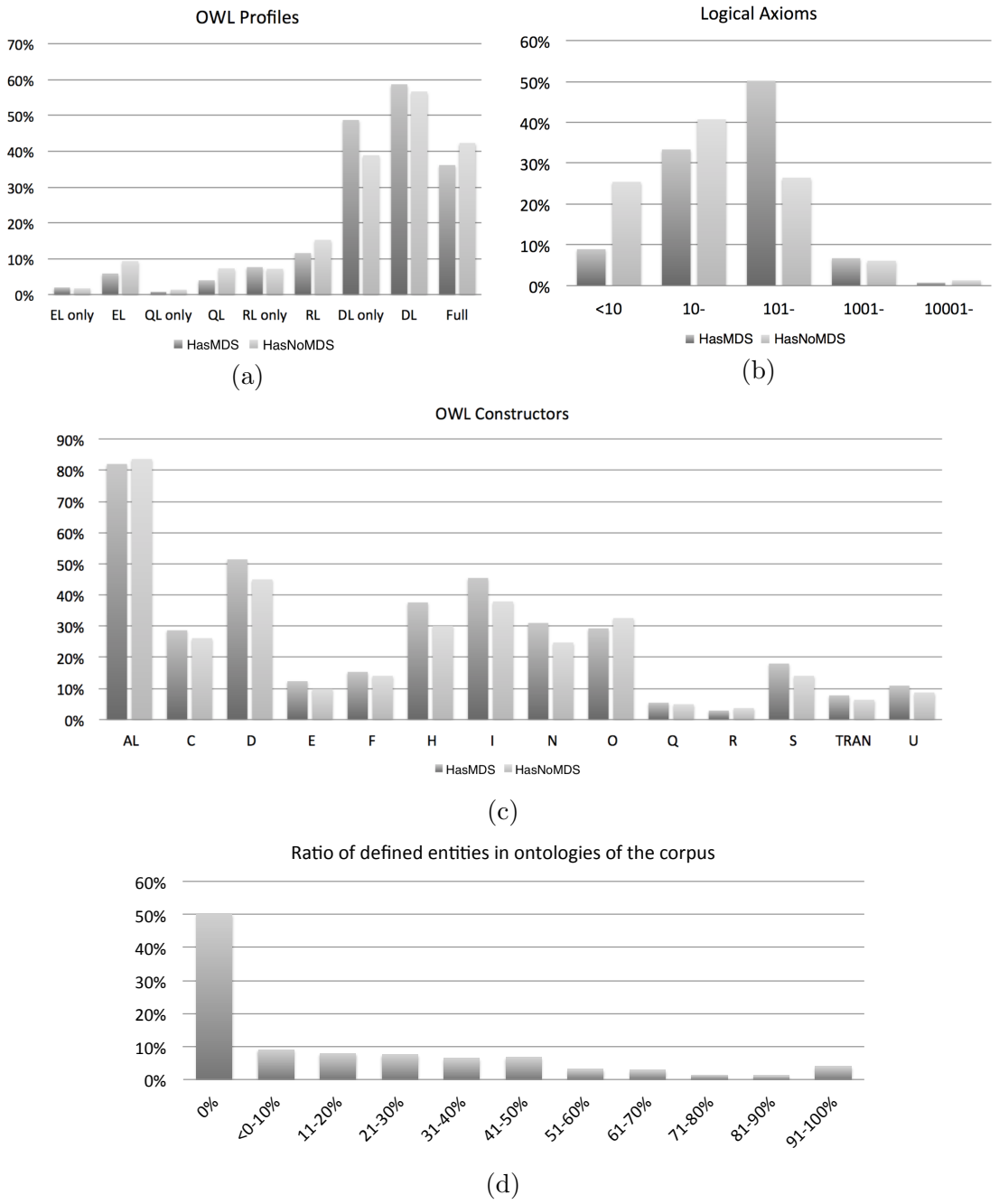


FIGURE 5.3: Comparing defined and undefined ontology properties

(5.3.c) presents the relative distribution of *HasMDS* and *HasNoMDS* ontologies, binned by the number of logical axioms; Figure (5.3.a) shows the distribution of OWL profiles in *HasMDS* and *HasNoMDS* ontologies; and Figure (5.3.b) shows the OWL constructor usage in *HasMDS* and *HasNoMDS* ontologies. Other than a small number of outliers, the results broadly demonstrate an even distribution of *HasMDS* and *HasNoMDS* ontologies, w.r.t. size, OWL profiles, and constructors. Thus, definability may occur in any type of ontology, regardless of the DL language employed, the size of an ontology, the conceptualised domain of interest, or its origin (i.e. source of creation). The only

ID	Ontology	$\mathcal{DL}$ Expressivity	Logical Axioms	$N_{\mathcal{C}}$ $\lvert N_{\mathcal{C}}\rvert$	$Def\%$	nb_M	$N_{\mathcal{R}}$ $\lvert N_{\mathcal{R}}\rvert$	$Def\%$	nb_M
Conference corpus									
1	*cmt*	$\mathcal{ALCIN}(\mathcal{D})$	226	29	13.79%	2.00	59	67.80%	1.00
2	*conference*	$\mathcal{ALCHIF}(\mathcal{D})$	285	59	49.15%	2.31	64	65.63%	1.00
3	*confOf*	$\mathcal{SIN}(\mathcal{D})$	196	38	18.42%	4.00	36	5.56%	1.00
4	*edas*	$\mathcal{ALCOIN}(\mathcal{D})$	739	103	11.65%	7.00	50	56.00%	1.00
5	*iasted*	$\mathcal{ALCIN}(\mathcal{D})$	358	140	11.43%	2.50	41	39.02%	1.00
6	*sigkdd*	$\mathcal{ALEI}(\mathcal{D})$	116	49	16.33%	2.38	28	42.86%	1.00
			320.00	**69.67**	**20.13%**	**3.36**	**46.33**	**46.14%**	**1.00**
LargeBio corpus									
7	*NCI_fma*	$\mathcal{ALC}$	9083	6488	30.27%	1.32	64	0.00%	0.00
8	*SNOMED_fma*	$\mathcal{ALER}$	20243	13412	21.44%	1.37	19	0.00%	0.00
9	*SNOMED_nci*	$\mathcal{ALER}$	71042	51128	57.31%	1.09	52	0.00%	0.00
			33456.00	**23676.00**	**36.34%**	**1.26**	**45.00**	**0.00%**	**0.00**

TABLE 5.5: Sample corpus properties, where $N_{\mathcal{C}}$ represents the set of concepts names, and $N_{\mathcal{R}}$ represents the set of role names.

property, which appears affects the level of definability in an ontology, is unsurprisingly, the granularity of conceptualisation.

5.2.2 Experiment 2: Definability Computation

These experiments investigate the feasibility of using the algorithms for computing definability, by analysing their behaviour and performance. The corpus used in this experiment consists of a variety of small ontologies that model the same domain (i.e. the Conference track within the OAEI corpus) and three large biomedical ontologies (LargeBio track, OAEI corpus). The aim of these experiments is to assess the time taken by the proposed approximation when computing MDS over a variety of ontologies, and to compare the performance of the single and the multi entity pruning approaches for MDS computation. Table 5.5 presents the characteristics of the *sample corpus*, including details of the DL expressivity, number of logical axioms, and the number of concept ($|N_{\mathcal{C}}|$) and role names ($|N_{\mathcal{R}}|$). Furthermore, the table shows the ratio of entities in the ontology signature that have MDSs ($Def\%$), and the average number of unique MDS per defined entity, given the complete set of MDSs (i.e. the result of the definability computation stage three), for both concepts and roles.

The Stages of Definability Computation. Table 5.6 is divided into three numbered partitions that show the time taken to compute the various steps (or stages) required in determining definability, where each step is measured in terms of the computation time (this is given either in seconds, or in hours in some cases), and the number of implicit definability checks (#Imp). The first stage establishes the definability status of each concept and role ($N_{\mathcal{C}} \cup N_{\mathcal{R}}$) in the ontology signature. In the second stage, the disjoint MDSs are computed, and then all MDSs are computed in the final stage. In both of the latter stages, nb_M denotes the MDSs to entity (with that has MDSs) ratio.

$\mathcal{O}$	(1) Definability Status ($N_C \cup N_R$)			(2) Disjoint MDSs ($N_C \cup N_R$)			(3) All MDSs ($N_C \cup N_R$)		
ID	Time	#Imp	$Def\%$	Time	#Imp	nb_M	Time	#Imp	nb_M
Conference corpus									
1	2.88s	176	51.16%	1.85s	99	1.09	0.54s	36	1.09
2	3.35s	246	65.14%	6.63s	352	1.13	307.54s	19833	1.54
3	2.92s	148	15.79%	2.67s	195	2.33	8.03s	573	3.33
4	20.84s	306	28.99%	28.63s	397	1.18	23.22h	1509274	2.80
5	16.14s	362	17.58%	150.89s	212	1.25	1058.74s	1756	1.75
6	3.22s	154	28.57%	2.55s	122	1.50	0.62s	61	1.55
	8.22s	**232.00**	**31.61%**	**32.50s**	**229.50**	**1.41**	**3.93h**	**255255.50**	**2.01**
LargeBio corpus									
7	5.97h	13104	29.98%	222.38h	229206	1.31	115.76h	118456	1.32
8	21.49h	26862	21.41%	234.75h	109980	1.09	756.53h	384930	1.37
9	392.95h	102360	57.25%	1885.39h	1145057	1.07	3991.85h	2619125	1.09
	140.14h	**71163.00**	**36.21%**	**780.84h**	**494747.53**	**1.16**	**1621.38h**	**1040837.00**	**1.26**

TABLE 5.6: *Cost measured in terms of different characteristics* (time, and number of definability checks $\#Imp$), and *results* ($Def\%$: definable entities in an ontology, nb_M: number of MDSs per entity that have MDSs) of the three stages of definability computation.

$\mathcal{O}$	Nb of Defined Concepts	**Single prune**	**Multi prune**		
		Avg. #Imp	Avg. #Imp	*SD*	*Single/Multi*
Conference corpus					
1	4	23.00	24.00	0.00	*0.1044*
2	29	254.00	258.12	4.36	*0.1038*
3	7	113.00	69.38	0.55	*0.5915*
4	12	1496.00	350.00	1.02	*0.2426*
5	15	1124.00	323.92	1.94	*0.2896*
6	8	44.00	46.00	0.00	*0.11364*
		509.00	**178.57**		***0.3508***

TABLE 5.7: Comparing single and multi-entity pruning MDS computation approaches.

In general, the larger, more expressive ontologies take much longer to compute than the smaller, less expressive ones. The definability status and the disjoint MDS computation stages are feasible for both small and large ontologies; whereas obtaining the complete set of MDSs (stage 3) is a considerably more costly operation. In most cases the MDS expansion (Alg. 5) is restricted to computing an MDS union size (see Section 4.6.1) $|\mathcal{S}| \leq 20$, which does not exclude any entities in the Conference, but excludes 441 entities in the LargeBio corpus). However, despite the computational cost incurred in computing the last stage, the difference between the number of MDSs found during stage 2 (on average 1.70 MDSs per entity with MDSs in the small, and 1.16 in the large ontologies) and 3 (2.41, and 1.25 MDSs, respectively), in many cases is negligible (0.71, and 0.09 MDSs more per entity). A notable case is the small *edas* ontology, where the first two stages take only 49.47 *seconds* to complete; however, the last stage takes 23.22 *hours*, although the MDSs to entities (that have MDSs) ratio has more than doubled (from 1.18 to 2.80). In the *confOf* ontology, the last stage also shows a significant increase, from 2.33 MDSs per entity to 3.33, but in this case the computation time is close (8.03 sec) to the sum of the two prior steps (5.59 sec).

Single and Multi-entity Pruning. This experiment compares the single entity (Algorithm 6) to the multi-entity pruning approach (which employs a divide and conquer strategy presented in Algorithm 7) in terms of the number of implicit definability checks performed while computing a single MDS of a given concept (that have MDSs). Although the former approach always uses linear number of calls to the oracle to find an MDS (i.e. the required number of definability checks always equal the size of the input signature, excluding the particular entity in question), the latter depends on the ratio of required and redundant signature member entities and their position in the given input signature. Thus, the process can differ w.r.t. the best and worst case time complexity. The experiment was conducted by computing an MDS for each concept of all small ontologies, and repeating the process 25 times, as multi-entity pruning involves a stochastic component, i.e. the entity ordering of the processed signatures.

Large ontologies require significantly longer time to compute due to the increased time of the implicit definability check performed on a larger set of axioms and signatures. Therefore, for these ontologies the MDS computation was *simulated*. Although the time taken by the implicit definability check method could vary between different entities, the real non-deterministic part of the MDS computation is the number of implicit definability checks performed by a given approach. As the MDSs were pre-computed, the outcome of the definability check could be determined without reasoning (i.e. making calls to the oracle). This meant that it was feasible to run the experiment for each concept with MDSs of the large ontology, and repeat 25 times.

Table 5.7 presents the results of the experiment. The middle partition shows the number of implicit definability checks performed by the single prune approach, while the right-hand side partition provides information about the latter approach, in terms of the mean number of definability checks, its standard deviation (SD), and the definability check ratio of the two approaches ($Single/Multi$). As expected, on average the latter approach performs better requiring only 35.73% of the number of definability checks conducted by the former approach. However, in 3 of 6 of the small ontologies (1,2 and 6), the latter performs worse than the former (in the worst case, the latter approach could require twice as many definability checks as the former i.e. $Single/Multi = 2.0$). This is explained by the fact that, in the case where small ontologies are used, the input signature is relatively small, i.e. it contains a higher ratio of required members. Table 5.12 presents the average number of input signature for the small (6.75, 9.76, 6.50 for ontology 1,2 and 6) and larger (17.14, 125.67, 71.31 for 3,4 and 5) ontologies.

Minimal Definition Signatures. Table 5.8 provides further information about the computed MDSs (for each ontology in the sample corpus) of concepts (top partition) and roles (bottom partition). The left partition shows the total number of entities in the ontology signature that have MDSs, the total number of unique MDSs all such entities in the ontology ($|M|$); the middle partition presents the number of MDSs per entity

$\mathcal{O}$	Nb of Defined Entities	$\|M\|$	MDSs per Defined Entity				MDS Cardinality			
			min	avg	med	max	min	avg	med	max
Concept MDSs										
1	4	8	1	2.00	2	3	1	1.38	1	3
2	29	67	1	2.31	1	14	1	2.75	3	5
3	7	28	1	4.00	4	8	1	1.46	1	2
4	12	84	1	7.00	1	34	1	3.74	4	6
5	16	40	1	2.50	2	5	1	1.95	2	5
6	8	19	1	2.38	2	6	1	1.21	1	2
	12.67	**41.00**	**1.00**	**3.36**	**2.00**	**11.67**	**1.00**	**2.08**	**2.00**	**3.83**
7	1964	2583	1	1.32	1	5	2	5.07	4	33
8	2879	3929	1	1.36	1	30	1	4.80	4	720
9	29460	31831	1	1.08	1	5	1	5.59	6	15
	11434.33	**12781.00**	**1.00**	**1.25**	**1.00**	**13.33**	**1.33**	**5.15**	**4.67**	**256.00**
Role MDSs										
1	40	40	1	1.00	1	1	1	1.00	1	1
2	42	42	1	1.00	1	1	1	1.00	1	1
3	2	2	1	1.00	1	1	1	1.00	1	1
4	28	28	1	1.00	1	1	1	1.00	1	1
5	16	16	1	1.00	1	1	1	1.00	1	1
6	12	12	1	1.00	1	1	1	1.00	1	1
	23.33	**23.33**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**

TABLE 5.8: Computed concept and role MDSs in the sample corpus.

that has MDSs (in terms of minimum, average, median and maximum); and the right partition describes the cardinality of MDSs.

MDS per concept scores show that, for small ontologies, about half of concepts that have MDSs have two MDSs, whereas for the large ontologies, most concepts have only one MDS, and there are few concepts in each ontologies with large number of MDSs. The average cardinality of a concept MDS is low, with 2.08 entities per MDS for small ontologies, and 5.15 for large ontologies. However, there are some extreme cases, such as the *SNO_nci* ontology, where one MDS contains 720 entities. None of the roles had MDSs in the large ontologies, and thus have been omitted from the table. However, for the small ontologies, every role had exactly one MDS. This is the result of that each of these roles, in the semantically rich and well-defined small ontologies, are simply implicitly definable either as a synonym or an inverse role, but no complex role definitions were used, as it can be seen by the average cardinality of the role MDSs.

Definition Patterns. As previously discussed (Section 3.4), a non-exhaustive list of concept and role definition patterns were identified. Table 5.9 presents the statistical evaluation of how often this pattern occurs in the ontologies used for empirical evaluation, it shows the distribution of concept and role MDSs w.r.t. to the corresponding definition patterns, where pattern numbers reference Table 5.10, *UnCls.* denotes those MDS cases that were unclassifiable under the identified patterns. Patterns that have no MDS in the sample corpus are omitted for brevity. Out of all MDSs in the Conference corpus only 23.88% of all cases in this corpus were unclassifiable, however, in the Large-Bio corpus 52.00% of all MDSs were unclassifiable. In the Conference corpus, 63.91% of all MDSs correspond to a single entity (i.e. where $|\text{MDS}| = 1$, these are patterns:

$\mathcal{O}$	Definition Patterns									
	Concept									Role
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**NoPat.**	**11**
Conference corpus										
1	9.68%	0.00%	0.00%	0.00%	29.03%	19.35%	3.23%	0.00%	0.00%	38.71%
2	2.08%	0.00%	0.00%	0.00%	8.33%	4.17%	0.00%	0.00%	2.08%	83.33%
3	0.00%	0.00%	0.00%	0.00%	46.67%	3.33%	0.00%	0.00%	43.33%	6.67%
4	7.34%	0.00%	0.00%	21.10%	5.50%	5.50%	0.92%	0.00%	21.10%	38.53%
5	26.79%	1.79%	7.14%	0.00%	0.00%	0.00%	21.43%	0.00%	14.29%	28.57%
6	2.68%	0.00%	0.00%	3.57%	3.57%	0.89%	0.89%	0.89%	62.50%	25.00%
	8.09%	**0.30%**	**1.19%**	**4.11%**	**15.52%**	**5.54%**	**4.41%**	**0.15%**	**23.88%**	**36.80%**
LargeBio corpus										
7	67.52%	0.00%	0.00%	22.45%	0.00%	0.00%	0.00%	0.00%	10.03%	0.00%
8	16.03%	0.00%	0.00%	17.54%	0.00%	0.00%	0.00%	0.00%	66.43%	0.00%
9	16.90%	0.00%	0.00%	15.53%	0.00%	0.00%	0.00%	0.00%	67.55%	0.00%
	33.48%	**0.00%**	**0.00%**	**18.51%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**48.00%**	**0.00%**

TABLE 5.9: Corresponding Definition Patterns of the MDSs in the sample corpus.

Concept					
1	Explicit definition	4	Disjoint union	7	Synonym role (domain or range)
2	Explicit synonym	5	Role domain concept	8	Inverse role (domain or range)
3	Implicit synonym	6	Role range concept		
Role					
9	Explicit definition	11	Explicit inverse	13	Implicit inverse
10	Explicit synonym	12	Implicit synonym		

TABLE 5.10: Definition Pattern Reference Guide.

$\mathcal{O}$	Nb of defined concepts	*Redundant concept*		*Implicitly Defined By Empty Signature*	*Unintended Synonyms*
Conference corpus					
1	4	0	*0.00%*	0	0
2	29	5	*17.24%*	0	0
3	7	0	*0.00%*	0	0
4	12	4	*33.33%*	0	0
5	16	0	*0.00%*	0	0
6	8	4	*50.00%*	0	0
	12.67	**2.17**	***17.11%***	**0.00**	**0.00**
LargeBio corpus					
7	1964	214	*10.90%*	0	0
8	2876	2295	*79.80%*	0	0
9	29300	24282	*82.87%*	0	0
	11380.00	**8930.33**	***78.47%***	**0.00**	**0.00**

TABLE 5.11: Ontology modelling errors in the sample corpus.

2, 3, 5-8, 11) definition pattern, as explained in later chapters MDS cardinality is an important factor in the cost of semantic interoperability (smaller MDSs are preferred to larger ones); whereas all definitions in the LargeBio corpus ontologies fall into the multi-entity pattern category.

Modelling Errors. Table 5.11 shows the occurrence of the three ontology modelling errors (Chapter 3.5) in the sample evaluation corpus. The only error, which can be found in these particular ontologies, concerns the phenomenon that explicit concept

definitions do not alway correspond to an MDS, as some entities of the signature may be redundant. This occurs frequently in the large biomedical ontologies, for instance, 79.80% and 82.87% of all defined concepts in the *SNO_fma* and *SNO_nci* ontologies exhibits this pattern, and therefore is not considered an error.

5.2.3 Experiment 3: Impact of Modularisation

This experiment set analyses the impact of modularisation to definability computation. Modularisation for definability computation involves producing a module which completely describes a given entity, i.e. the subject matter of the module has always one member, the entity in question. A module is typically a smaller subset of the TBox, the first experiment compares TBox-es and modules in terms of number of axioms and signature cardinality. Intuitively, as modules are typically smaller then TBox-es, the time taken by an implicit definability check performed on a module is expected to be faster than when a definability check is conducted on a whole TBox; thus the second experiment evaluates this claim.

Axioms and Signatures. Table 5.12 compares the size of the axiom set ($\mathsf{Ax}(\mathcal{T})$ and $\mathsf{Ax}(\mathcal{M})$) and the signature ($\mathsf{Sig}(\mathcal{T})$ and $\mathsf{Sig}(\mathcal{M})$) of a TBox ($\mathcal{T}$), and a syntactic-locality-based star $\mathcal{S}$-module of a single entity ($\mathcal{M}$), for every entity in all ontologies. The table is divided into four parts, showing the results separately for concepts with and without MDSs, and roles (large ontologies in the sample corpus contained no roles with MDSs, thus these are omitted). In general, single entity $\mathcal{S}$-modules are significantly smaller than the corresponding TBox, both in terms of number of axioms and signature cardinality; for the concepts with MDSs of the small ontologies, on average the axiom set is only 21.31%, and the signature is 27.37% of the whole TBox. Furthermore, modules of entities without MDSs are typically smaller than modules of entities with MDSs, with 9.21% and 18.16% average axiom set and signature size, respectively.

Role modules are the product of modularising the RBox of an ontology, which is typically much smaller than the TBox, thus modularisation provides a less sizeable reduction for roles. Moreover, modules of roles without MDSs often have empty axiom sets and signatures, but a module of a role with MDS is never an empty set. This is a useful heuristic for definability status computation, as the implicit definability check is not necessary to be performed for empty modules.

Definability Check. This experiment compares the time taken by the *implicit definability check* method in a TBox or in a module. In every ontology of the sample corpus, for each all concepts and roles, the experiment measured the time of the implicit definability check (i.e. is the given entity implicitly definable in an ontology, where the input signature is the entire ontology or module signature, excluding the entity name), performed in a TBox and in a corresponding module. In addition, the experiment measured the modularisation time. Table 5.13 presents the results of the experiment, where

$\mathcal{O}$	$\lvert\mathsf{Ax}(\mathcal{T})\rvert$	$\lvert\mathsf{Ax}(\mathcal{M})\rvert$					$\lvert\mathsf{Sig}(\mathcal{T})\rvert$	$\lvert\mathsf{Sig}(\mathcal{M})\rvert$				
		min	avg		med	max		min	avg		med	max
***Concepts** with MDSs*												
1	226	6	14.25	*6.31%*	15.50	20	95	4	6.75	*7.11%*	7.00	9
2	285	7	27.00	*9.47%*	23.00	59	125	3	9.76	*7.81%*	8.00	19
3	196	33	64.14	*32.73%*	55.00	111	78	11	17.14	*21.98%*	15.00	26
4	739	129	147.33	*19.94%*	135.50	191	157	120	125.67	*80.04%*	122.50	139
5	358	3	168.38	*47.03%*	167.50	223	182	2	71.31	*39.18%*	71.50	93
6	116	4	14.38	*12.39%*	12.00	44	80	3	6.50	*8.13%*	5.00	17
	320.00	**30.33**	**72.58**	***21.31%***	**68.08**	**108.00**	**119.50**	**23.83**	**39.52**	***27.37%***	**38.17**	**50.50**
7	9083	9	8940.78	*98.43%*	12366.00	16264	6542	5	1719.65	*26.29%*	2390.00	3085
8	20243	7	132.81	*0.66%*	125.00	476	13429	5	45.09	*0.34%*	43.00	149
9	71042	7	175.63	*0.25%*	172.00	757	51179	5	59.35	*0.12%*	59.00	243
	33456.00	**7.67**	**3083.08**	***33.11%***	**4221.00**	**5832.33**	**23716.67**	**5.00**	**608.03**	***8.91%***	**830.67**	**1159.00**
***Concepts** without MDSs*												
1	226	1	2.96	*1.31%*	1.00	22	95	1	1.92	*2.02%*	1.00	9
2	285	1	5.27	*1.85%*	1.00	36	125	1	2.40	*1.92%*	1.00	13
3	196	1	29.00	*14.80%*	36.00	61	78	1	9.29	*11.91%*	12.00	17
4	739	116	121.46	*16.44%*	120.00	193	157	115	117.62	*74.91%*	117.00	140
5	358	1	71.60	*20.00%*	1.00	223	182	1	30.84	*16.94%*	1.00	93
6	116	1	1.00	*0.86%*	1.00	1	80	1	1.00	*1.25%*	1.00	1
	320.00	**20.17**	**38.55**	***9.21%***	**26.67**	**89.33**	**119.50**	**20.00**	**27.18**	***18.16%***	**22.17**	**45.50**
7	9083	2	44.61	*0.49%*	2.00	15086	6542	3.00	11.32	*0.17%*	3.00	2880
8	20243	2	2.01	*0.01%*	2.00	12	13429	3	3.00	*0.02%*	3.00	6
9	71042	2	3.07	*0.00%*	2.00	339	51179	3	3.34	*0.01%*	3.00	106
	33456.00	**2.00**	**16.56**	***0.17%***	**2.00**	**5145.67**	**23716.67**	**3.00**	**5.89**	***0.07%***	**3.00**	**997.33**
***Roles** with MDSs*												
1	29	1	1.40	*4.83%*	1.00	3	41	2	2.00	*4.88%*	2.00	2
2	46	1	1.52	*3.31%*	2.00	2	46	2	2.00	*4.35%*	2.00	2
3	3	1	1.00	*33.33%*	1.00	1	4	2	2.00	*50.00%*	2.00	2
4	14	1	1.00	*7.14%*	1.00	1	28	2	2.00	*7.14%*	2.00	2
5	8	1	1.00	*12.50%*	1.00	1	16	2	2.00	*12.50%*	2.00	2
6	6	1	1.00	*16.67%*	1.00	1	12	2	2.00	*16.67%*	2.00	2
	17.67	**1.00**	**1.15**	***12.96%***	**1.17**	**1.50**	**24.50**	**2.00**	**2.00**	***0.16***	**2.00**	**2.00**
***Roles** without MDSs*												
1	29	0	0.05	*0.18%*	0.00	1	41	0	0.05	*0.13%*	0.00	1
2	46	0	0.05	*0.10%*	0.00	1	46	0	0.05	*0.10%*	0.00	1
3	3	0	0.06	*1.96%*	0.00	1	4	0	0.06	*1.47%*	0.00	1
4	14	0	0.00	*0.00%*	0.00	0	28	0	0.00	*0.00%*	0.00	0
5	8	0	0.00	*0.00%*	0.00	0	16	0	0.00	*0.00%*	0.00	0
6	6	0	0.00	*0.00%*	0.00	0	12	0	0.00	*0.00%*	0.00	0
	17.67	**0.00**	**0.03**	***0.37%***	**0.00**	**0.50**	**24.50**	**0.00**	**0.03**	***0.28%***	**0.00**	**0.50**
7	0	0	0.00	*0.00%*	0.00	0	0	0	0.00	*0.00%*	0.00	0
8	2	0	0.00	*0.00%*	0.00	0	4	0	0.00	*0.00%*	0.00	0
9	10	0	0.00	*0.00%*	0.00	0	15	0	0.00	*0.00%*	0.00	0
	4.00	**0.00**	**0.00**	***0.00%***	**0.00**	**0.00**	**6.33**	**0.00**	**0.00**	***0.00%***	**0.00**	**0.00**

TABLE 5.12: Comparing the size, in terms of *number of axioms* ($\lvert\mathsf{Ax}(\mathcal{T})\rvert$ and $\lvert\mathsf{Ax}(\mathcal{M})\rvert$) and *signature cardinality* ($\lvert\mathsf{Sig}(\mathcal{T})\rvert$ and $\lvert\mathsf{Sig}(\mathcal{M})\rvert$), of a TBox and an $\mathcal{S}$-module of a single concept or role.

O	*Concepts*					*Roles*				
	$\lvert N_C\rvert$	$\mathsf{Imp}(\mathcal{T}\text{-Box})$	*Mod.*	$\mathsf{Imp}(\mathcal{M})$	*Total*	$\lvert N_\mathcal{R}\rvert$	$\mathsf{Imp}(\mathcal{R}\text{-Box})$	*Mod.*	$\mathsf{Imp}(\mathcal{M})$	*Total*
Conference corpus										
1	29	20.23ms	0.59ms	14.83ms	*76.20%*	59	31.83ms	0.35ms	29.09ms	*92.51%*
2	59	22.85ms	0.70ms	18.09ms	*82.22%*	64	74.29ms	1.15ms	48.22ms	*66.45%*
3	38	26.02ms	0.78ms	24.01ms	*95.31%*	36	50.27ms	0.13ms	37.00ms	*73.87%*
4	103	122.64ms	1.76ms	72.94ms	*60.91%*	50	105.28ms	0.17ms	100.36ms	*95.49%*
5	140	70.31ms	1.09ms	33.44ms	*49.11%*	41	46.29ms	0.12ms	42.83ms	*92.81%*
6	49	23.65ms	0.70ms	17.67ms	*77.65%*	28	14.09ms	0.11ms	18.52ms	*132.21%*
	69.67	**47.62ms**	**0.94ms**	**30.16ms**	***73.57%***	**46.33**	**53.67ms**	**0.34ms**	**46.00ms**	***92.22%***
LargeBio corpus										
7	6488	8.08s	0.39s	8.97s	*115.9%*	64	2.77s	0.01ms	2.68s	*96.97%*
8	13412	10.62s	0.48s	12.49s	*122.1%*	19	4.68s	0.01ms	4.64s	*99.18%*
9	51128	54.74s	2.39s	46.60s	*89.5%*	52	35.88s	0.01ms	43.42s	*120.99%*
	23676.00	**24.48s**	**1.09s**	**22.69s**	***109.16%***	**45.00**	**14.44s**	**0.01ms**	**16.91s**	***105.71%***

TABLE 5.13: Comparing the time taken by the implicit definability check method, performed in a TBox and an $\mathcal{S}$-module of a single concept or role.

$\mathsf{Imp}(\mathcal{T}/\mathcal{M})$ denotes the average time taken to complete the implicit definability check (for all concepts or roles of a given ontology), *Mod.* denotes the average time required to extract a module, and *Total* denotes the ratio of the modularised implicit definability check time (which is the sum of the module extraction, and the definability check time) and the definability check time performed on a TBox.

Both for concepts and roles, in small and large ontologies, modularisation considerably reduces the time taken by the implicit definability check method. In the small ontologies, on average the definability check takes 26.43% less time to complete; this is due the fact that, as previously demonstrated modules are typically smaller than whole TBox-es, furthermore, the module extraction time is negligible compared to the implicit definability check time. Thus modularisation is indeed an effective mechanism for reducing the complexity of definability computation.

5.3 Summary and Conclusions

This chapter have presented an empirical analysis about the prevalence and the extent of definability over a wide range of OWL ontologies, and the practical applicability of the proposed definability computation algorithms over a sample dataset. The empirical results described in this chapter suggest that:

- Implicit definability may occur in any type of ontology, regardless of the employed DL language, the size of an ontology, the conceptualised domain of interest, or its origin (source of creation); although it is more likely to occur in more expressive, and semantically richer ontologies. Therefore the exploitation of MDSs could indeed benefit semantic interoperability. As implicit definability check works in languages that do not accept Beth definability, the experiment corpus was not filtered by checking which ontology language accept Beth definability.

- Establishing the definability status of a given concept or role is a feasible process for most real-world ontologies. Computing a set of disjoint MDS is also feasible, however in general, the larger, more expressive ontologies take much longer to compute than the smaller, less expressive ones.

- Obtaining the complete set of MDSs is feasible for most smaller ontologies, but it could take a considerable amount of time for larger, or more expressive ontologies. However, as the definability computation process is parallelizable (i.e. it can be performed separately for each entity of a given subject matter), it can be potentially feasible to compute for larger ontologies as well.

- Modularisation is an effective mechanism for reducing the complexity of definability computation, as it reduces the size of the input, and the axiom set of the definability computation algorithms, resulting in significant time reduction in the

performance of the implicit definability check. The actual positive effect of modularisation is multiplied in MDS computation, as such algorithms are implemented by conducting repeated definability checks.

Part III

Application to Ontology Alignment

Chapter 6

Ontology Alignment

Software components often need to reconcile knowledge represented in different ontological models, where entities can vary in the labels used to identify them or in their logical definitions through one or more axioms: that is these ontologies are *heterogeneous*. The problem of Ontology Matching (often also called Ontology Alignment) addresses the challenge of reconciling ontology heterogeneity by establishing logical correspondences between entities in different ontologies in order to support ontology interoperability, thus forming an *alignment*: i.e. a set of ontological correspondences that map entities from one ontology to those corresponding entities in another ontology. This chapter provides the background for (*i*) Chapter 8, which explores the applications of definability and minimal definition signatures in the context of ontology alignment and alignment negotiation, and shows that MDSs entail a new type of *definability-based correspondences*, which are often non-simple complex correspondences that are only found by semantic matching approaches. It also presents an overview of the different *ontology alignment evaluation* techniques that provide the foundations for the *definability-based ontology alignment evaluation*. Moreover this chapter underpins (*ii*) the motivation of this thesis, *ontology alignment negotiation* and (in Section 9.2) outlines an approach that exploits MDSs and the notion of minimal signature coverage to improve semantic interoperability between knowledge-based agents. This chapter surveys the vast area of ontology alignment with a specific focus only on the relevant notions that support the definitions and the findings presented in the remaining chapters of this thesis. Sections 6.1 define ontology heterogeneity and the types of mismatches that can affect heterogeneous ontologies, while Section 6.2 introduces the foundations of ontology alignment. Section 6.3 focusses on the state of the art in semantic ontology matching methods, that make use of logics to verify the correctness of mappings, and often repair them, and can be used to define new correspondences. Section 6.4 describes ontology alignment evaluation models and measures that were the basis for the empirical evaluation framework used in Chapter 8. Lastly, Section 6.5 reviews ontology alignment negotiation, which explores how heterogeneous knowledge-based systems can cooperate in identifying an alignment through negotiation, and surveys the notable alignment negotiation approaches, as this

was meant to be the primary area for the applicability of the main contribution of this thesis.

6.1 Ontology Heterogeneity

Chapter 2 introduced the notion of ontologies as a computing artefact and the role they play in knowledge representation and reasoning by making explicit and formalising knowledge about constraints and assumptions. The formalisation of knowledge in an explicit and machine readable format is fundamental to support *sharing* and *interoperability* amongst different computational systems developed independently from each others [68]. This notion is the premise for the Semantic Web [11] and Linked Open Data [71], that have promoted the adoption of ontology based representation. However, the success and uptake of these areas meant that distinct systems cannot be assumed to adhere to the same ontologies, even when representing the same domain. Knowledge sharing, and in general semantic interoperability is inherently hindered or even precluded between distinct ontologies, as independently designed ontologies typically introduce syntactically and semantically different domain conceptualisations.

Heterogeneity between independently developed ontologies can manifest itself in different ways, and a number of efforts have attempted to classify the different types of heterogeneity. Gómez-Pérez [64] was amongst the first researchers to classify the problems that might be encountered when knowledge is to be shared. She proposed two main categories:

1. Heterogeneity problems;
    (a) *Heterogeneity of knowledge representation formalism*;
    (b) *Heterogeneity of the implementation languages*;
    (c) *Lexical problem* (also identified as *implicit inconsistencies problem* in [102]);
    (d) *Synonymity*;
2. Background assumptions problems;
    (a) *Hidden assumptions*;
    (b) *Loss of common sense knowledge.*

Kitakami distinguished between *non-semantic* and *semantic heterogeneity* [85]. Non-semantic heterogeneity broadly corresponds both to the lexical problem and to the heterogeneity of the implementation language identified by Gómez-Pérez. Non-semantic heterogeneity is also described by Visser *et al.* and Klein who call it called *syntactic* or *language heterogeneity* respectively in [146] and [86]. *Semantic heterogeneity*, also called *ontology heterogeneity* by Visser and colleagues [146], occurs when different assumptions are made regarding the conceptualisation of the same domain, which are represented in the ontology. This is a type of heterogeneity that typically occurs when attempting to

integrate ontologies representing partially overlapping domains. Ontology heterogeneity manifests itself in two types of mismatches, namely *conceptualisation* and *explication mismatches*, which refer to two different stages in constructing an ontology: the *conceptualisation* of a domain and its explicit representation or *explication* [1]. Broadly speaking *conceptualisation mismatches* are semantic differences that are due to different conceptualisations of the classes and relationships in the domain being modelled. They include model coverage and granularity and scope, i.e. differences in the extensions of two classes that are meant to be the same. *Explication mismatches* are typically due to differences in the way a domain is specified, i.e. in the choice of *terms* used to label a term, and in the formal *definition* associated with the label. Klein [86] further defined the types of conceptualisation and explication mismatches, and reviewed and classified the technique used to overcome each of the identified types of heterogeneity. Building on this body of work, Euzenat [43, 47] used the classification of heterogeneity types to classify the matching techniques used to address the types of mismatches identified by Visser and Klein. Euzenat [47] classifies matching techniques across two dimensions:

- Classification of matching techniques based on the input to the matching approach, i.e. the types of objects that are manipulated by the matching technique.
- Classification of matching techniques based on the interpretation of the input information.

From this latter perspective, Euzenat defines semantic matching as the class of techniques that use some formal semantics in order to interpret the input and provide some form of justification of the obtained match. Semantic matching techniques appear in both directions and are based on well founded deductive methods. If two elements of an ontology are matched semantically then they should have the same interpretation. The contribution in this thesis aims to explore implicit definability to support this type of matching of ontological resources.

6.2 Ontology Alignment

Ontology matching (also called alignment) addresses the fundamental and ubiquitous issue of ontology heterogeneity by producing *alignments*, i.e. sets of *correspondences* that describe the logical relations (i.e. $\equiv, \sqsubseteq, \sqsupseteq, \perp$ etc.) between semantically related entities of different ontologies [106]. Ontology matching has become an established research field, and received increasing interest in the past two decades [47, 106, 124]. In a recent and comprehensive literature review, Otero-Cerdeira *et al.* [106] reported that between 2003 and 2013, 694[2] articles were published in the following areas: reviews, matching techniques and systems, practical frameworks and applications, and evaluation.

[1] These two phases are named following Gruber's definition of an ontology as an "explicit specification of a conceptualisation" [68]

[2] This number represents only those publications where ontology matching was the main focus. The total number of papers, related to ontology matching, was over 1600.

Given two ontologies $\mathcal{O}$ and $\mathcal{O}'$, an alignment A is the set of *correspondences* between $\mathcal{O}$ and $\mathcal{O}'$.

Definition 6.1 (Correspondence [47])**.** A correspondence is a triple $c = \langle \mathsf{e}, \mathsf{e}', r \rangle$, asserting that some relation $r \in \{\equiv, \sqsubseteq, \sqsupseteq, \perp\}$ holds between entities $\mathsf{e} \in \mathcal{O}$ and $\mathsf{e}' \in \mathcal{O}'$.

Note that in literature (and in this thesis), the term *"mapping"* is often used as a synonym word for ontological *correspondence*.

For example, one ontology may define a large vehicle for transporting goods as a Lorry, while another may refer to the same concept as a Truck. In order to reconcile this terminological heterogeneity, ideal matching systems would produce the correspondence $\langle \mathsf{Truck}, \mathsf{Lorry}, \equiv \rangle$, which describes the two concepts as interchangeable synonym words under the ontologies.

In addition to identifying the relation existing between two terms belonging to two different ontologies, a correspondence may also contain some *metadata*, which is assigned by the matcher system that produces the correspondence. The most common additional information attached to a correspondence is the *confidence measure*; i.e. a correspondence could also be defined as a 4-tuple $c = \langle \mathsf{e}, \mathsf{e}', r, cf \rangle$, where and cf denotes the confidence score. The confidence measure is a real number within the range $[0, 1]$, expressing the normalised degree of trust regarding the correctness of the correspondence, such that higher values denote higher confidence, of the producing matcher. The confidence measure is often utilised both internally and externally. For example, a matcher may use a threshold parameter to exclude any correspondence from the final alignment, where the confidence degree is below some threshold. Furthermore, when there are several different alignments present for a given ontology pair, or there are more than one correspondences available for a given entity, the confidence measure may be used as a base of comparison between the ambiguous correspondences (i.e. correspondences that align the same entities).

Based on the classification of heterogeneity proposed by Visser [146], and later by Klein [86] and Euzenat [43], Euzenat and Shvaiko [47] surveyed the different approaches for aligning ontologies, and classified them on the grounds of the interoperation of the input information:

- **Semantic:** These methods utilise model-theoretic semantics to determine whether or not there is a correspondence between two entities, and hence are typically deductive. Such methods may include propositional satisfiability and modal satisfiability techniques, or logic based techniques.

- **Internal Structural:** Methods for determining the similarity of two entities based on the internal structure, which may use criteria such as the range of their properties (attributes and relations), their cardinality, and the transitivity and/or symmetry of their properties to calculate the similarity between them.

- **External Structural:** Methods for determining external structure similarity may evaluate the position of the two entities within the ontological hierarchy, as well as comparing parent, sibling or child concepts.
- **Terminological:** These methods lexically compare the strings (tokens or n-grams) used in naming entities, or in the labels and comments concerning entities. Such methods may employ normalisation techniques (often found in Information Retrieval systems) such as stemming or eliminating stop-words, etc.
- **Extensional:** Extension-based methods which compare the extension of classes, i.e., their set of instances. Such methods may include determining whether or not the two entities share common instances, or may use alternate similarity based extension comparison metrics.

Here we mention the different types of alignments for completeness, and we focus on *semantic approaches* since they allow us to exploit the contribution of this thesis, i.e. the notion of *definability* for the problem of aligning ontologies.

Definition 6.1 defines the notion of a *simple correspondence*, i.e. a relation that exists between single entities. However, some relations cannot be expressed by using simple correspondences, for example, the correspondence $\langle \mathsf{Father}, (\exists.\mathsf{hasChild} \sqcap \mathsf{Man}), \equiv \rangle$ maps a simple concept of one ontology to a complex concept of another ontology; moreover the correspondence $\langle (\mathsf{Mother} \sqcup \mathsf{Father}), (\exists.\mathsf{hasChild}), \equiv \rangle$ maps two complex concepts. This is often referred to as a *complex correspondence* [47] and defined as follows:

Definition 6.2 (Complex correspondence)**.** Given two ontologies $\mathcal{O}$ and $\mathcal{O}'$, a complex correspondence is a triple $\mathsf{c} = \langle \mathsf{C}, \mathsf{C}', \mathsf{r} \rangle$, asserting that some relation $\mathsf{r} \in \{\equiv, \sqcap, \sqcup, \bot\}$ holds between C and C', where

- either C is a concept name, and C' is a complex concept, such that $\mathsf{C} \in \mathcal{O}$ and $\mathsf{Sig}(\mathsf{C}') \subseteq \mathsf{Sig}(\mathcal{O}')$;
- or C' is a concept name, and C is a concept definition, such that $\mathsf{C}' \in \mathcal{O}'$ and $\mathsf{Sig}(\mathsf{C}) \subseteq \mathsf{Sig}(\mathcal{O})$;
- both C' and C are complex concepts, such that $\mathsf{Sig}(\mathsf{C}) \subseteq \mathsf{Sig}(\mathcal{O})$ and $\mathsf{Sig}(\mathsf{C}') \subseteq \mathsf{Sig}(\mathcal{O}')$.

Therefore, the definition of alignment can be extended to including both simple and complex correspondences:

Definition 6.3 (Alignment [47])**.** Given two ontologies $\mathcal{O}$ and $\mathcal{O}'$, an alignment A between $\mathcal{O}$ and $\mathcal{O}'$ is a set of (simple or complex) correspondences between pairs of entities, or complex concepts belonging to $\mathcal{O}$ and $\mathcal{O}'$.

6.3 Semantic Ontology Matching

This section provides an overview of the state-of-the-art of *semantic ontology matching* that relate to the work presented in Chapter 8, which introduces definability-based correspondences. This is not an exhaustive overview, but an outline of the approaches that have influenced the work presented in this thesis.

Despite the fact that numerous matching approaches have been developed in the past two decades, in recent field surveys, Shvaiko and Euzenat [47, 124] note the *lack of expressive alignments*. Most matching systems focus on finding simple, non-oriented (i.e. equivalence) correspondences that match concepts to concepts, or roles to roles, and only a few approaches are able to identify complex, directed (i.e. specifically from ontology $\mathcal{O}$ to ontology $\mathcal{O}'$, and therefore not symmetric) correspondences.

In 2003, Bouquet *et al.* introduced the first semantic matching system, *CtxMatch* [18], that treats ontology matching as a logical validity problem, and determines equivalence and subsumption mapping by employing DL reasoners. *S-Match*, introduced by Giunchiglia in 2004 [61], was initially an extension of CtxMatch. The system was further developed by Giunchiglia, who presented [59] an open source semantic matching framework that transformed data structures such as business catalogs, conceptual models and web services descriptions into lightweight ontologies, and provided an extensible API for developing new algorithms. Recently, Giunchiglia and colleagues [60] described a matching technique based on S-Match, which computed the minimal mapping set for lightweight ontologies (i.e. an alignment that is irreducible without losing the property that any other mappings can be efficiently computed from the minimal set).

In 2003, Borgida and Serafini introduced Distributed Description Logics [13] (DDLs). DDLs formally represent modular ontologies as distributed T-Boxes, where each module is a different T-Box, pairwise interrelated to others by *bridge rules*. A bridge rule expresses links between concepts of different ontologies in the form $\mathcal{O}_i : \mathsf{A} \xrightarrow{r} \mathcal{O}_j : \mathsf{X}$, where $R \in \{\equiv, \sqsubseteq, \sqsupseteq, \bot\}$, stating that a concept in $\mathcal{O}_i$ is in some (set theoretical) relation with a concept in $\mathcal{O}_j$. The conjunction of mappings $\mathcal{O}_i : \mathsf{A} \xrightarrow{\sqsupseteq} \mathcal{O}_j : \mathsf{X}$ and $\mathcal{O}_i : \mathsf{A} \xrightarrow{\sqsubseteq} \mathcal{O}_j : \mathsf{X}$ is equivalent to the mapping $\mathcal{O}_i : \mathsf{A} \xrightarrow{\equiv} \mathcal{O}_j : \mathsf{X}$. Bridge rules are directional, the stated semantic relation only applies from one viewpoint. Building on bridge rules, in 2005 Serafini *et al.* [122] proposed a distributed reasoning architecture (DRAGO) that address the problem of reasoning in a network of (interconnected) ontologies. The authors shown that bridge rules transfer knowledge thus enable *knowledge propagation*. In this thesis, bridge rules and distributed reasoning were the inspiration for the notion of aggregated correspondences and reconciliation of mapping relations in definability-based correspondences, which enables the propagation of implicit definability (Section 8.1). Extending the semantics of DDLs, Atencia *et al.* [3] introduced formal semantics for *weighted mappings*, that make use of the weight (i.e. the confidence value) assigned to mappings by the producing matching system. The authors note that distributed mappings are uncertain due to nature of alignment (which is a "best guess" by a matcher

based on matching techniques asserting that some semantic relation holds between entities of different ontologies) therefore they use reclassification semantics (i.e. given a set of instance of the aligned concepts, if transferred between the aligned ontologies, what portion of the instances would be reclassified under the pairwise mapped concept) to assign an upper and lower bound value (between a range of 0 and 1) to the mappings, e.g. $\mathcal{O}_1 : \mathsf{Truck} \xrightarrow{\equiv_{[0.75,0.90]}} \mathcal{O}_2 : \mathsf{Lorry}$.

In 2004, Dhamankar *et al.* presented one of the first complex matching approaches, *iMAP* [35], which semi-automatically created simple and complex mappings between relational database schemas. As the search space of complex mappings is possibly unbounded, iMAP employs a search mechanism which focuses on the 'meaningful' parts of the space, using information about the domain to guide the search process.

Scharffe *et al.* introduced and later extended a library of *correspondence patterns*, that by capturing "regularities recurring when aligning ontologies" aids in modelling ontology alignments in terms of simple and complex correspondences [119–121]. In 2009, Ritze *et al.* proposed a pattern-based approach to detect complex correspondences [112], by exploiting an initial simple alignment. As noted by the authors, the set of non-exhaustive patterns only covers a part of the otherwise infinite search space of candidate complex correspondences. The original approach was extended in 2010 by additional patterns, and by integrating natural language processing techniques [113]. The approach presented by Svab-Zamazal *et al.* (2010) [133] builds on Scharffe's correspondence patterns, by applying entity and axiom level transformations based on the mutual occurrence of patterns in the aligned ontologies, to yield complex correspondences.

In 2010, Stuckenschmidt *et al.* argued that the application of machine learning techniques such as Inductive Logic Programming is a natural fit for complex matching, as complex correspondences are analogous to complex logical rules [131]. This has inspired a concrete approach that was presented and empirically evaluated by Hu *et al.* in 2011 [79]. The approach described by Jiang *et al.* [80] (2015) used a probabilistic framework which created knowledge-rules, based on the axiomatisation of the ontology, in order to restrict the infinite search space of complex mappings.

Jimenez-Ruiz *et al.* introduced *LogMap* [81], a highly scalable matching system which is suitable to align large, semantically rich ontologies. In 2012 the approach was extended to LogMap2 [82], in order to facilitate user interaction to the matching process. LogMap uses anchoring: first it computes a set of equivalence correspondences using basic matching methods, then from the anchored classes, it examines the extended class hierarchy to discover further simple mappings that may be oriented (expressing subsumption, or disjointness relations). LogMap is able to find non-equivalence correspondences. Furthermore, the approach uses reasoning to identify and repair those mappings that cause inconsistency or incoherence in the merged ontology. The notion of alignment incoherence and alignment repair was also investigated by Meilicke *et al.* [100]. This approach does not produce alignments itself but uses the semantics of alignments and ontologies to diagnose *minimal conflict sets*, i.e. the smallest possible set of erroneous mappings

whose removal from the alignment makes it coherent. Based on Meilicke's degree of measuring alignment incoherence, the OAEI now assesses not only the precision and recall of the evaluated matching approaches, but their consistency and incoherence [65].

The *STROMA* system, introduced in 2014 by Arnold and Rahm [1, 2], applied an enrichment strategy to an initial alignment, provided by the popular matching system COMA 3.0 [94]. The strategy exploited linguistic techniques and background knowledge to determine subsumption relations.

The notion of bridge rules, of mapping repairs, and much of the work overviewed above was the inspiration for applying implicit definability to ontology alignment. As implicit definability permits defined entities to be removed from an alignment without semantic loss, implicit definability entails a new type of *definability-based (or implicit) correspondence*, based on the definition signatures of entities in the aligned ontologies and the available alignment. Such correspondences:

i are justified by model-theoretic semantics;

ii typically take the form of a complex correspondence;

iii are consequences of aligned ontologies;

iv are instantiated by anchoring on existing correspondences;

v are composed of several simple mappings;

vi are created whilst avoiding the generation of erroneous correspondences by applying filtering to existing (anchor) correspondences in order to maintain consistency of both the alignment and the merged ontology (which is the union of two pairwise aligned ontologies and a corresponding alignment).

The approach presented in Chapter 8 does not generate an alignment, but through reasoning, it aggregates an existing correspondence set to find (by inference) "missing" correspondences, or to strengthen the initial alignment. Hence, this approach can be validly considered as a *semantic ontology matching technique*.

6.4 Ontology Alignment Evaluation

Over the past decade, the ontology matching field has seen an increasing interest from the research community. A significant number of studies have focused on developing new, or improved matching techniques and alignment systems. The most recent, comprehensive literature review of the ontology matching field notes that between 2003 and 2013, over 85 articles were published describing more than 50 different matching systems [106]. However, so far no 'silver bullet' has been found (i.e. no approach can be singled out as the best solution) to address the potentially complex and multi-faceted heterogeneity problem. There are matching systems that only perform well in particular areas (e.g. instance matching, large-scale matching etc.), whereas others may be usable for a wide

range of matching tasks. In order to assess the systems performance, they need to be evaluated.

The Ontology Alignment Evaluation Initiative (OAEI) is a coordinated international initiative concerned with improving the work on ontology matching by assessing strengths and weaknesses of alignment systems, comparing performance of techniques, improving evaluation techniques, and increasing communication among algorithm developers [46]. The datasets used for evaluation are a combination of synthetic and real world ontologies, for instance those contributed by the Ontofarm project [152] [3] that have been adapted to support the evaluation of different aspects of alignment approaches.

Since 2004, the OAEI has carried out comprehensive evaluation campaigns and publishes its results on a yearly basis. Evaluation datasets and results are available on the OAEI website[4].

The standard metrics used to measure the efficiency of alignment systems were adopted from the information retrieval field, and include *precision*, *recall* and *f-measure* metrics [47]. *Precision* measures the degree of correctness, whereas *recall* measures the degree of completeness of a produced alignment. The *f-measure* is the harmonic mean of precision and recall. These metrics are calculated by comparing the output of a particular matching system (i.e. the produced alignment) to a reference alignment; these are then used for evaluation, for example by the OAEI to rank the alignment systems. All three metrics provide a real number within the range $[0, 1]$. The definition of these metrics are given as follows:

Definition 6.4 (Precision, recall and f-measure [47])**.** Given a produced alignment A and a reference alignment R,

- the *precision* of A with respect to R is defined as $pr(A, R) = \frac{|A \cap R|}{|A|}$
- the *recall* of A with respect to R is defined as $rc(A, R) = \frac{|A \cap R|}{|R|}$
- the *f-measure* of A with respect to R is defined as $f_\alpha(A, R) = (1 + \alpha^2) \cdot \frac{pr \cdot rc}{\alpha^2 \cdot pr + rc}$ such that for $F.5$ the $\alpha = 0.5$, for $F1$ the $\alpha = 1$, and for $F2$ the $\alpha = 2$.

6.4.1 Semantic Precision and Recall

Over the past decade, the well-understood precision and recall metric have undergone several revisions to overcome a variety of limitations. Euzenat *et al.* [44] introduced *semantic precision and recall* to address the issue that semantically equivalent, but syntactically different alignments are potentially assigned different precision and recall scores. This approach compares the semantic closure of alignments, i.e. the complete set of correspondences, which is entailed by the union of the ontologies merged by the corresponding alignment, whereas its syntactic variants consider only the explicitly stated

[3] OntoFarm is a collection of heterogeneously structured ontologies describing the same domain, that of conference organisation.

[4] `http://oaei.ontologymatching.org`

correspondences. Fleischhacker *et al.* implemented and tested this metric family [50], and concluded that alignments had to be restricted to simple mappings, as considering complex alignments results in an infinite set of entailments (semantic alignment closure).

Euzenat *et al.* revised [28], and subsequently implemented [30] semantic precision and recall as part of the Alignment API. However, it was noted, that despite the revisions, this metric family does not: provide an absolute measure (such as that obtained by the classical precision and recall metrics); satisfy all desired properties of evaluation metrics; or accommodate complex mappings[5]. The OAEI continues to use the syntactic precision and recall metrics, although they also compare produced alignments with an entailed reference alignment, which is generated as the transitive closure computed on the original reference alignment [23].

6.4.2 Similarity in the Alignment Space

Classical ontology distance and proximity measures [27] deal with computing proximity or similarity between ontologies by directly comparing their content (i.e. the evaluation is carried out in the *ontology space*). David *et al.* [31] have introduced a new family of metrics that measure ontology similarity in the *alignment space* (a network of ontologies connected by alignments) by evaluating the similarity between two ontologies with regard to the set of available alignments between them. In contrast with the use of the syntactic precision and recall metrics, an alignment is not evaluated against a reference alignment, but assessed with respect to a given signature to establish the quality of the provided coverage. Furthermore, David *et al.* explored the notion of alignment paths, i.e. sequence of alignments between two ontologies in an alignment space. These measures are useful for alignment evaluation in scenarios when the aligned ontologies are not available (e.g. private), or when similarity must denote the ability to transfer information within a network of discrete knowledge-based systems, for example a Multi-Agent System (MAS) [150].

Moreover, David *et al.* tested both the metrics computed in the ontology, and the alignment space in order to determine the proximity between a set of pairwise aligned ontologies used for creating and characterising peers, and their connections in a semantic *social network* [29]. They argued that similarity measures can indicate the social affinity of peers, i.e. closer peers are more likely to engage in meaningful interactions.

Cerqueus *et al.* characterised and measured the level of *semantic heterogeneity* in peer-to-peer systems by defining a *disparity metric*, which expresses the ratio of unmapped entities of an ontologies signature [21]. As noted by the authors, disparity is the inverse of the coverage metric (introduced in [31]).

Locoro *et al.* employed the aforementioned ontology [27] and alignment space [31] based metrics in order to rank candidate intermediate ontologies and their corresponding

[5]However, it is worth to mention that few existing matching systems are currently able to produce complex correspondences.

available alignments, that provide contextual background information for their semantic ontology matching approach.

6.5 Ontology Alignment Negotiation

In the past decade, *Ontology Alignment Negotiation* has became an established and active research area, whereby several studies have explored how heterogeneous knowledge-based systems can cooperate in identifying an alignment through negotiation [47, 124, 137]. *Knowledge-based agents* are autonomous programs that model their world and internal preferences using ontologies [8]. Agents with shared or overlapping ontologies can reason over, and exchange both terminological and assertional knowledge and ultimately communicate in terms of concept definitions or instances. However, several challenges have emerged, which derive from the dynamic, heterogeneous nature of such multi-agent systems. Agents can significantly vary with respect to their goals, preferences and origin; these properties often cannot be anticipated or planned for at design-time. Moreover, different agents use different vocabularies for describing the same or partially similar domains. In order to be able to carry out meaningful communication in dynamic and opportunistic scenarios (e.g. in e-commerce, open-data or mobile systems), independently designed agents need to reach a mutual understanding of the entities (concepts, roles etc.) in the exchanged messages or agree on a common subset of concept translation correspondences. Hence vocabularies need to be reconciled, by *cooperatively establishing a mutually acceptable alignment*. Assuming a set of existing alignments that are either stored from previous interactions, or computed on the fly, some form of *decentralised negotiation* is carried out via an argumentation or a dialogue framework. Thus alignment negotiation can be considered as an alignment "aggregation technique" [47]. The remainder of this chapter briefly reviews the notable approaches and the state of the art of the field.

In 2002, Bailin and Truszkowski coined the term *Ontology Negotiation* (between intelligent information agents) [8]. By building on the notion of *ontology exchange* (whereby ontological information is exchanged via messages) and human conversation model patterns (such as seeking clarification by providing further references or information), Bailin and Truszkowski provide an automated protocol that allows agents to establish a mutually acceptable alignment through incremental and recursive interpretation, clarification and explanation of the exchanged messages. However, their approach only considers the use of equivalence correspondences. In the same year, Bouquet *et al.* approached the (same) problem of *"meaning negotiation"* by developing a theoretical framework and a concrete language (*ConTeXt Markup Language*) for describing agents' ontologies and mappings between them [16]. This language was succeeded by C-OWL, which permits the expression of semantic correspondences between heterogeneous OWL ontologies [17].

Silva *et al.* presented (2005) an *"ontology mapping negotiation"* approach whereby agents carry out quantitative negotiation about correspondences. This negotiation was

based based on the combination of *utility and meta-utility functions that assess the confidence* of the correctness and applicability of correspondences (given agents' local context and preferences), ultimately leading to a decision, i.e. accepted, rejected, or negotiated [125]. The approach is highly constrained as correspondences are produced by *one* alignment system (MAFRA [93]), thus it cannot be flexibly applied to other environments.

Laera *et al.* proposed (2006) a *Meaning-based Argumentation* (MbA) framework to facilitate agents arguing over ontology alignments in MAS [89–92]. The approach exploits Bench-Capon"s Value-based Argument Framework [9] (or VAF, which assigns different strengths to arguments) by distinguishing between successful and failed attacks on the possible acceptance (or replacement of) candidate correspondences, based on local knowledge and the agent's preferences over the justification for accepting the correspondences. Correspondences can be categorised by the method in which they are generated, such as the Terminological, Structural, Extensional, etc. (presented earlier in Section 6.2). A *candidate correspondence* (i.e. a mapping under consideration) is the subject of arguments, whereby agents assess mappings and justify their choice (accept or reject) according to their internal preferences and ontological knowledge. Furthermore, the approach is compatible with a wide range of matching systems that are suitable for aligning OWL ontologies, and that formalised their result using the Alignment Format.

Doran *et al.* introduced (2009) a *flexible approach for determining agents orientation on ontology mappings* (FDO) [38, 39, 42]. This approach also exploited the VAF-based correspondence negotiation approach, however, it provided a more flexible framework than Laera's MbA approach, by permitting agents to express a minimum acceptability thresholds for each of the mapping type. Furthermore, Doran *et al.* explored the use of modularization to as a space reduction mechanism for ontology negotiation [40, 41]. Alignment negotiation can be viewed as essentially a bilateral search process, whereby agents exhaustively explore all possible correspondences. However, as the search process is potentially computationally costly (at the worst case using the VAF $\Pi_2^{(p)}$ [40]), modularization can be used a pre-processing step to reduce the number of correspondences that are considered by the argumentation process.

Trojahn *et al.* (2010) proposed a Strength-based Argumentation Framework (SVAF) that (similarly to MbA and FDO) exploited Bench-Capon's VAF, by permitting arguments to be represented with a strength value, which reflected the confidence score that was assigned by the matching system producing the alignment [138]. Moreover, Trojahn *et al.* presented an additional negotiation approach, where ontology mappings are represented as disjunctive OWL-DL queries [139].

The argumentation approach presented by Spiliopoulos *et al.* employed the max-sum algorithm for synthesising ontology alignment methods [128]. In their work, *"each agent is responsible for computing mappings concerning a specific ontology element, using a specific alignment method"*. The approach does not satisfy privacy constraints, as agents have complete knowledge of the aligned ontologies.

Atencia and Schorlemmer presented an interaction-based approach to semantic alignment [4]. The approach adheres to the notion of *emergent semantics* (Chapter 1.4.2 [47]), whereby alignments are not generated by matching systems but learned from an agents interaction experience. As the agents continuously revise and negotiate the meaning of ontological terms, there is the possibility that they may never arrive to a final state (i.e. a total consensus). In a similar approach, proposed by Chocron and Schorlemmer [24], a pre-existing alignment is assumed, and agents repair alignments using information learned during the interaction process.

Payne and Tamma proposed (2014) [108] and later significantly extended [109, 110] the *Correspondence Inclusion Dialogue* (CID), which allows agents to exchange communicative acts (assert, counter, accept and reject) regarding to the available correspondences in a shared alignment space. In a CID, agents negotiate the acceptability of candidate correspondences by exchanging the *degree of beliefs* (i.e. weight) associated with correspondences. As there could be multiple alignments providing alternative choices of correspondences, the selection is based on the combined, or joint weight of both interacting agents. Agents strive to minimise the number of disclosed (and thus negotiated) correspondences in order to minimise the disclosure of their ontological knowledge as well as to reduce the overall cost of the negotiation. The resulting mutual alignment is always unambiguous, i.e. each aligned ontology entity is covered by a single correspondence.

The work presented by Jimenez-Ruiz *et al.* provides a refinement of the CID, by limiting consistency and conservativity violations that could occur [83, 84]. The conservativity principle states that the aligned ontology should not cause any change in the agents' local ontologies [127]. The approach uses the LogMap matching system to detect those correspondences whose acceptance would introduce violations and to perform alignment repair.

Santos *et al.* presented a cognitive-speech based dialogue protocol to support meaning negotiation between knowledge-based agents [116, 117]. In contrast to traditional negotiation approach, the agents possess no prior knowledge of existing alignments; just their respective ontologies. The agents thus take turns in exchange details of those entities deemed pertinent to the construction of an alignment in order to support a given task (i.e. align its signature). The agent processing the proposal then determines semantic similarity by employing basic alignment techniques (e.g. lexical similarity), on the fly.

Chapter 7

Minimal Signature Coverage

This chapter introduces and empirically evaluates a novel algorithm, which by exploiting a priori obtained complete set of minimal definition signatures, efficiently produces an approximation of the smallest possible entity combination, the *minimal signature cover*, that provides coverage for a given entity set (a subject matter). The presented algorithm is employed in Chapter 8, and could be employed in ontology alignment negotiation, where minimal signature cover sets contribute to improving semantic interoperability by providing optimised coverage for tasks and by reducing the cost of establishing alignments between knowledge-based agents.

As previously discussed, ontology signatures can describe more than the asserted concept, role and individual names because an entity which has a MDS can be removed without semantic loss as the meaning of the definable entity is wholly fixed by the terms of its definition. Therefore, a given signature may facilitate the expression of not only its constituent entities, but also those definable entities whose definition is permitted with the given signature. A carefully composed signature may support and enhance a variety of semantic interoperability scenarios, in particular ontology matching, where typically incomplete alignments provide partial coverage for an ontology vocabulary, i.e. only a *restricted signature* is available to support semantic interoperability. Moreover, in ontology alignment negotiation [83, 110, 116], where an alignment is required to be mutually acceptable for all interacting parties and it is the product of a negotiation process, it is beneficial to *minimise* the considered correspondences (i.e. the aligned part of an ontology signature) in order to reduce the overall cost of the process, and to support emerging constraints (privacy, confidentiality, etc.).

F. van Harmelen and colleagues [142] have shown that semantic interoperability tasks can be characterised (amongst other parameters) by their signature, thus the prerequisite for performing a knowledge-based task is that the tasks's signature must be *covered* by the terms available to the party that performs the given task. In order to determine whether a given *task signature* is covered by a given *restricted signature*, each entity of the task signature must be individually examined; an entity is covered either if it appears in the restricted signature, or if it has an MDS using only the members of the restricted signature. Although task coverage is trivial to establish, determining the

minimal signature that covers a given task signature poses a challenge, as the complete set of MDSs needs to be known, and all combinations of such definition signatures are required to be explored, for each entity in question.

Please note that in all algorithms presented in this chapter we consider the reasoning task of deciding implicit definability, which is dedicated to an external reasoner system, as constant time single step computation, i.e. treating them as **oracle calls** [107]. Each algorithm complexity analysis applies just to the complexity of the algorithm in question and not to the combined running time of the algorithm and the implicit definability check(s) performed by the oracle.

The remainder of this chapter is organised as follows: Section 7.1 discusses the set coverage problem family, which relates closely to the signature coverage problem, and as such has inspired the presented approach. Section 7.2 introduces and characterises the signature coverage problem, while Section 7.3 presents the approximation algorithm that produces minimal cover sets. Section 7.4 reports on the empirical evaluation, including the experiment framework, methodology and the results. Section 7.5 summarises and concludes the chapter.

7.1 The Set Coverage Problem Family

In this section we review two classical problems that address the issue of coverage: the set coverage problem and the minimal functional dependency cover. The *set coverage problem* (or minimal set cover problem) is a classic problem in combinatorics, complexity theory and computer science, in general [145].

Let $\mathbf{U}$ be a set of elements (referred to as the *universe*) and $\mathbf{S}$ a collection of subsets of $\mathcal{U}$, whose union equals the universe. The set cover problem identifies the smallest, minimal sub-collection $\mathbf{C} \subseteq \mathbf{S}$, called the cover set, such that the union of sets in $\mathbf{C}$ covers $\mathbf{U}$ (i.e. $\forall x\{x \in \mathbf{U} | x \in \mathbf{C}\}$). For instance, let us consider Example 7.1:

Example 7.1 (Minimal set cover problem)**.** *Consider the set* $\mathbf{U} = \{1, 2, 3, 4, 5\}$*, and let* $\mathbf{S}$ *be the collection* $\{\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_4\}$*, where* $\mathbf{s}_1 = \{1, 2, 3\}, \mathbf{s}_2 = \{1, 2\}, \mathbf{s}_3 = \{3, 4\}$ *and* $\mathbf{s}_4 = \{4, 5\}$*. The union of subsets of* $\mathbf{S}$ *contains all members of* $\mathbf{U}$*, thus* $\mathbf{U}$ *can be covered by an* $\mathbf{S}' \subseteq \mathbf{S}$*. Although there are a number of possible solutions, there is only one minimal cover set,* $\mathbf{s}_1, \mathbf{s}_4$*.*

Finding the minimal cover set is an NP-complete problem, however, there is a greedy algorithm that is able to find *approximations* (i.e. not necessarily minimal, but small cover sets) in polynomial time [145].

In the *weighted set cover problem*, each set $\mathbf{s}_i \in \mathbf{S}$ is assigned a weight $w(\mathbf{s}_i) \geq 0$, and in this case, the goal is to find a cover set $\mathbf{C}$ with the minimal total weight $\sum_{\mathbf{s}_i \in \mathbf{C}} w(\mathbf{s}_i)$ (where the weight of a set does not correspond to its cardinality but emerges from the particular context where the set is used).

Functional Dependency Coverage. Another classical problem that has influenced

the approach presented in this paper comes from relational database theory, and concerns finding the minimal functional dependency cover. A *functional dependency (FD)* is a constraint between two sets of attributes in a relation from a database [144]. For instance, given two attribute sets X and Y , then the FD $X \rightarrow Y$ means that the values of the attribute set Y are determined by the values of X, or in other words, two tuples in a database sharing the same values of X would also share the same values for Y. The closure of a set of attributes X with respect to a set of FDs $\mathbf{F}$ is the set X^+ of all attributes that are functionally determined by X using the closure of $\mathbf{F}$, denoted by $\mathbf{F}^+$. Before computing the closure, a set of FDs $\mathbf{F}$ is usually normalised by exhaustively applying inference rules (e.g. reflexivity, transitivity and augmentation; Table 7.1 presents the *inference rules* [144] that are used both in normalisation and in closure computation (where X, Y, Z, W denote attribute sets in some relation R).

Inference Rule	**Condition**	**Action**
Reflexivity	if $Y \subseteq X$	then $X \rightarrow Y$
Augmentation	if $X \rightarrow Y$	then $XZ \rightarrow YZ$
Transitivity	if $X \rightarrow Y$ and $Y \rightarrow Z$	then $X \rightarrow Z$
Union	if $X \rightarrow Y$ and $X \rightarrow Z$	then $X \rightarrow YZ$
Decomposition	if $X \rightarrow YZ$	then $X \rightarrow Y$ and $X \rightarrow Z$
Pseudotransitivity	if $X \rightarrow Y$ and $WY \rightarrow Z$	then $WX \rightarrow Z$
Composition	if $X \rightarrow Y$ and $Z \rightarrow W$	then $XZ \rightarrow YW$

TABLE 7.1: Functional dependency inference rules.

The following example illustrates how the closure of all attributes is computed by repeatedly applying inference rules:

Example 7.2 (FD set attribute closures)**.** *Let us consider a set of FDs* $\mathbf{F}$ *such that*

$$\begin{aligned}\mathbf{F} = \{ & \textit{(1)}\ A \rightarrow B \\ & \textit{(2)}\ C \rightarrow E \\ & \textit{(3)}\ E \rightarrow F \\ & \textit{(4)}\ A, C \rightarrow D\}\end{aligned}$$

$\mathcal{F}$ *is already normalised (in the third normal form). The closure of all attributes in* $\mathbf{F}$ *is computed as shown by the following steps:*

1. $A^+ : A, B$ *(A by reflexivity, B by (1))*
2. $B^+ : B$ *(B by reflexivity)*
3. $C^+ : C, E, F$ *(C by reflexivity, E by (2), F by transitivity and (2, 3))*
4. $D^+ : D$ *(D by reflexivity)*
5. $F^+ : F$ *(F by reflexivity)*

6. $(A, C)^+ : A, B, C, D, E, F$ *(A, C by reflexivity, B by (1), D by (4), E by (2), F by transitivity and (2, 3))*

The closure of a set of attributes with respect to a set of FDs allows us to determine the minimal cover for a set of functional dependencies. A set of functional dependencies $\mathbf{F}$ covers another set of FDs $\mathbf{G}$ if every functional dependency in $\mathbf{G}$ can be inferred from $\mathbf{F}$, i.e. if $\mathbf{G}^+ \subseteq \mathbf{F}^+$. $\mathbf{F}$ is a minimal cover of $\mathbf{G}$ if $\mathbf{F}$ is the smallest set of functional dependencies that covers $\mathbf{G}$. It can be proven that every set of functional dependencies has a *minimal cover*, however this is not unique, as there may be more than one minimal cover.

7.2 The Signature Coverage Problem

The ontology signature coverage problem concerns whether a given *task signature* $\mathfrak{S}$ can be covered by another, *restricted signature* $\mathfrak{R}$, where both signatures are subsets of the same ontology signature. A task signature is said to be covered if all of its constituent entities are covered. Individual names of a signature can only be covered by an asserted entity i.e. *explicitly*, however, definable signature entities (concepts and roles) can also be *covered implicitly* if the restricted signature contains a corresponding definition signature. We define entity coverage as:

Definition 7.1 (Explicitly or implicitly covered entity)**.** Given an ontology $\mathcal{O}$, a task signature $\mathfrak{S}$, and restricted signature $\mathfrak{R}$ such that $\mathfrak{S}, \mathfrak{R} \subseteq \mathsf{Sig}(\mathcal{O})$, an entity $\mathsf{e} \in \mathfrak{S}$

- is *covered explicitly* by $\mathfrak{R}$, if and only if $\mathsf{e} \in \mathfrak{R}$;
- or *covered implicitly* by $\mathfrak{R}$, if and only if e has a valid definition signature Σ under $\mathcal{O}$ such that $\Sigma \subseteq \mathfrak{R}$;

otherwise e is *uncovered* by $\mathfrak{R}$.

A *definable concept or role can simultaneously be covered explicitly and implicitly*, thus a task signature entity $\mathsf{e} \in \mathfrak{S}$ may assume one of the four different *coverage status* w.r.t. a restricted signature $\mathfrak{R}$; Table 7.2 lists the coverage statuses.

Coverability			Required Coverage
uncoverable		$\mathsf{e} \notin \mathfrak{R} \wedge \Sigma \not\subseteq \mathfrak{R}$	explicit coverage: $\mathsf{e} \in_? \mathfrak{R}$
coverable	explicitly only	$\mathsf{e} \in \mathfrak{R} \wedge \Sigma \not\subseteq \mathfrak{R}$	
	explicitly and implicitly	$\mathsf{e} \in \mathfrak{R} \wedge \Sigma \subseteq \mathfrak{R}$	explicit and implicit coverage: $\mathsf{e} \in_? \mathfrak{R}^+$
	implicitly only	$\mathsf{e} \notin \mathfrak{R} \wedge \Sigma \subseteq \mathfrak{R}$	

Table 7.2: Entity coverage statuses.

Determining whether a given task signature is coverable by a particular, restricted signature is the trivial process of identifying the coverage status of each task signature entity. This can be achieved in two ways:

1. either each task signature entity is subjected to an implicit definability check;

2. or by assuming that the complete set of MDSs of each entity is already obtained, for each entity $\mathsf{e}_i \in \mathfrak{S}$, we search for a corresponding MDS Σ^{e_i} such that $\Sigma^{\mathsf{e}_i} \subseteq \mathfrak{R}$.

The set of entities, which covers all members of a task signature $\mathfrak{S}$ is called the *cover set* $\mathfrak{C}$ and it is defined as follows:

Definition 7.2 (Cover set)**.** Given an ontology $\mathcal{O}$, a task signature $\mathfrak{S}$, and restricted signature $\mathfrak{R}$ such that $\mathfrak{S}, \mathfrak{R} \subseteq \mathsf{Sig}(\mathcal{O})$, $\mathfrak{C}$ is a cover set of $\mathfrak{S}$ with respect to $\mathfrak{R}$, if and only if

- $\mathfrak{C} \subseteq \mathfrak{R}$;
- $\forall \mathsf{e}\{\mathsf{e} \in \mathfrak{S} | \mathsf{e} \in \mathfrak{C} \vee \exists \Sigma^{\mathsf{e}} | \Sigma^{\mathsf{e}} \subseteq \mathfrak{C}\}$.

In other words, a cover set is a definition signature of all concepts and role names contained in the task signature.

As an ontology signature can cover more than its constituent entities, due to the fact that a given signature may permits some defined entities to be implicitly covered, we adopt the notion of *closure* from functional dependency computation (Section 7.1) in order to provide a signature representation which describes *the set of all entities covered* by a given signature. This is referred to as the signature closure, and defined as follows:

Definition 7.3 (Signature closure)**.** Given an ontology $\mathcal{O}$, and a signature $\mathcal{X}$ such that $\mathfrak{X} \subseteq \mathsf{Sig}(\mathcal{O})$, $\mathfrak{X}^+$ (the closure of $\mathfrak{X}$), contains all explicitly and implicitly covered entities of $\mathsf{Sig}(\mathcal{O})$ by $\mathfrak{X}$, i.e. the set of entities $\forall \mathsf{e}\{\mathsf{e} \in \mathsf{Sig}(\mathcal{O}) | \mathsf{e} \in \mathfrak{X} \vee \exists \Sigma^{\mathsf{e}} | \Sigma^{\mathsf{e}} \subseteq \mathfrak{C}\}$.

This permits a more succinct definition of signature coverage: a task signature $\mathfrak{S}$ is covered by a set $\mathfrak{C}$ if and only if $\mathfrak{S} \subseteq \mathfrak{C}^+$.

Once it has been established, that a restricted signature $\mathfrak{R}$ covers a given task signature $\mathfrak{S}$, the problem is to identify the *smallest subset* $\mathcal{C} \subseteq \mathfrak{R}$ which covers $\mathfrak{S}$. This is called the minimal cover set (or cover), and defined as follows:

Definition 7.4 (Minimal cover set)**.** Given an ontology $\mathcal{O}$, a task signature $\mathfrak{S}$, a restricted signature $\mathfrak{R}$ such that $\mathfrak{S}, \mathfrak{R} \subseteq \mathsf{Sig}(\mathcal{O})$, and the set $\mathfrak{C}$ which covers $\mathfrak{S}$ with respect to $\mathfrak{R}$, $\mathfrak{C}$ is minimal if and only if there is no other cover set $\mathfrak{C}' \subseteq \mathfrak{R}$ such that $|\mathfrak{C}'| < |\mathfrak{C}|$.

There can be more than one unique minimal cover set for a given task signature, i.e. two sets with the same cardinality score whose overlap is not an empty set.

Finding a minimal cover set is a high complexity problem, because it requires all cover sets to be identified by exhaustively testing each subset of the power set of the ontology signature ($\mathcal{P}(\mathsf{Sig}(\mathcal{O}))$), in order to find all covers and select the one with the smallest cardinality. This complexity can be reduced by considering the module of a given task signature, instead of the entire ontology signature. As it was shown in the previous chapters, all possible MDSs of a definable entity is contained in a module of the entity, thus the module of a task signature contains all possible minimal cover sets.

However, finding the minimal cover set in a module is still a high complexity problem, as in this case each subset of the power set of the module signature needs to be considered.

Approximation algorithms are commonly used for problems with non polynomial time complexity, such as the set cover problem, in order to provide sub-optimal solutions in polynomial-time. The *greedy algorithm design* is one of the standard techniques for approximation algorithms [145]. In the context of the minimal signature cover problem, the optimal solution is the smallest possible cover set. The following example illustrates the signature coverage problem:

Example 7.3 (Signature Coverage)**.** *Let $\mathcal{O}$ be an ontology, $\mathfrak{S}$ a task signature, $\mathfrak{R}$ a restricted signature, and $\mathbf{M}$ the complete set of MDSs of each definable entity of $\mathfrak{S}$ (an MDS $m \in \mathbf{M}$ is represented as the tuple $\langle e, \Sigma \rangle$, where e stands for the definable concept or role name, and Σ denotes the minimal definition signature), where $\mathfrak{S}, \mathfrak{R} \subseteq \mathsf{Sig}(\mathcal{O})$, such that*

- $\mathsf{Sig}(\mathcal{O}) = \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}, \mathsf{F}, \mathsf{r}, \mathsf{s}, \mathsf{q}\}$
- $\mathbf{M} = \{\langle \mathsf{C}, \{\mathsf{A}, \mathsf{B}\}\rangle, \langle \mathsf{C}, \{\mathsf{E}, \mathsf{r}\}\rangle, \langle \mathsf{C}, \{\mathsf{q}\}\rangle, \langle \mathsf{B}, \{\mathsf{D}\}\rangle, \langle \mathsf{D}, \{\mathsf{B}\}\rangle, \langle \mathsf{s}, \{\mathsf{r}\}\rangle\}$
- $\mathfrak{S} = \{\mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}, \mathsf{s}, \mathsf{q}\}$
- $\mathfrak{R} = \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}, \mathsf{r}, \mathsf{q}\}$

Without accounting for definability, i.e. by only considering explicit coverage, the restricted signature does not cover the task signature because $\mathfrak{S} \setminus \mathfrak{R} \neq \emptyset$. However, considering implicit coverage shows that the closure of the restricted signature is $\mathfrak{R}^+ = \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}, \mathsf{r}, \mathsf{s}, \mathsf{q}\}$, thus $\mathfrak{S}$ can be covered by $\mathfrak{R}$, because $\mathfrak{S} \subseteq \mathfrak{R}^+$.

Following a naive, greedy approach, one may select those entities that appear both in $\mathfrak{S}$ and $\mathfrak{R}$ as such entities can be covered explicitly, i.e. $\mathfrak{C}_1 = \mathfrak{S} \cap \mathfrak{R} = \{\mathsf{C}, \mathsf{D}, \mathsf{E}, \mathsf{q}\}$; then attempt to cover the remaining task signature entities, by adding a corresponding MDSs for each uncovered task signature entity; in this case role s is coverable by adding r to $\mathfrak{C}_1$, as there is an MDS $\langle \mathsf{s}, \{\mathsf{r}\}\rangle \in \mathbf{M}$, thus s is implicitly definable by the signature $\{\mathsf{r}\}$. As a result $\mathfrak{C}_1 = \{\mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}, \mathsf{r}, \mathsf{q}\}$ covers $\mathfrak{S}$. However, the smallest cover set is $\mathfrak{C}_2 = \{\mathsf{B}, \mathsf{E}, \mathsf{r}, \mathsf{q}\}$ as its closure is $\mathfrak{C}_2^+ = \{\mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}, \mathsf{r}, \mathsf{s}, \mathsf{q}\}$, therefore $\mathfrak{S} \subseteq \mathfrak{C}_2^+$, and $|\mathfrak{C}_1| > |\mathfrak{C}_2|$.

In example 7.3, a naive, greedy approach (*Greedy #1*) has produced a non-optimal cover set $\mathfrak{C}_1$, which was an approximation of the minimal cover $\mathfrak{C}_2$. The cover set $\mathfrak{C}_1$ can be improved by removing redundant entities, resulting in the set $\mathfrak{C}_1' = \mathfrak{C}_2$, hence producing a non-redundant cover set:

Definition 7.5 (Non-redundant cover set)**.** Given an ontology $\mathcal{O}$, a task signature $\mathfrak{S}$, a restricted signature $\mathfrak{R}$ such that $\mathfrak{S}, \mathfrak{R} \subseteq \mathsf{Sig}(\mathcal{O})$, and the set $\mathfrak{C}$ which covers $\mathfrak{S}$ with respect to $\mathfrak{R}$, $\mathfrak{C}$ is a non-redundant cover set if and only if none of its no proper subsets are cover sets of $\mathfrak{S}$.

Non-redundant cover sets are typically small, however, as there can be more than one non-redundant cover set (with different cardinality), a non-redundant cover set is not necessarily minimal. It is worth noting that every minimal cover set is also a non-redundant set.

7.3 Approximating Minimal Cover Sets

In order to tackle the complexity of the minimal signature cover problem, we introduce a greedy, approximation algorithm (*Greedy #2*), which provides a sub-optimal solution in polynomial-time; where the resulting cover set is always non-redundant. Although this problem is potentially easier then the reasoning task of deciding implicit definability (which includes the potentially very high computational complexity entailment check reasoning task, e.g. in the DL $\mathcal{SHROIQ}$ it is NP-hard), we note that the latter is dedicated to an oracle (i.e. we do not attempt to tackle that problem as it is out of the scope of this thesis), while the former problem is addressed by this work because in oppose to entailment checking, there is no such system which would provide a solution for minimal signature coverage (i.e. it is not a common reasoning task such as entailment check).

The basic idea behind the approach is that, starting from an empty set, the cover set is built up incrementally until all task signature members are covered, however, instead of selecting individual entities from the restricted signature, at each iteration the approach selects an entity set. The entity sets that are being considered are MDSs, because individual entities are typically only cover entities explicitly, while MDSs can cover definable entities implicitly (in addition to explicitly covering all those task signature entities that appear in the MDS as well). The selection is made by assigning a *cost* and *value* score to each MDS, and then picking the MDS which provides the maximum value and the minimum cost with respect to the task signature and the incomplete cover set, prioritising on the value score. The cost quantifies the number of entities required to be added to the cover set (i.e. the set difference of the cover set and the particular MDS), while the value represents the number of entities that the given signature covers (a given MDS can be a definition signature for more than one definable entity, thus it can cover several task signature entities). In case there are more than one MDSs with the same cost and value, a random MDS is selected.

In order to evaluate the *actual value* of a given MDS, i.e. the set of all entities of the task signature that the MDS covers either explicitly or implicitly, similarly to functional dependencies, its *closure* needs to be identified, thus we represent MDSs in the form of FDs to facilitate this notion. There is a strong resemblance between the concept of an FD and an MDS, meaning that an MDS can be thought of as functional dependency between entities of an ontology, where the relation between the signature of the left-hand side and the entity on the right-hand side is implicit definability. For example, the minimal definition signature $\Sigma^{\mathsf{C}} = \{\mathsf{A}, \mathsf{B}\}$ which defines concept C using

entities $\{\mathsf{A}, \mathsf{B}\}$ may be represented as $\mathfrak{m} : (\mathsf{A}, \mathsf{B} \rightarrow \mathsf{C})$; such an MDS is referred to as an fMDS, and it is defined as follows:

Definition 7.6 (fMDS)**.** Given a definable entity e, and its minimal definition signature Σ where $\mathsf{e} \in \mathsf{Sig}(\mathcal{O})$ and $\Sigma \subseteq \mathsf{Sig}(\mathcal{O})$, the corresponding fMDS is the function

$$\mathfrak{m} : (\Sigma \rightarrow \mathsf{e})$$

which given the entity set Σ implicitly covers the entity e.

Similarly to the closure of functional dependencies, the closure of an fMDS is computed from the *set of all fMDSs*, by identifying all relevant definition signatures and repeatedly applying inference rules:

Definition 7.7 (fMDS-closure)**.** Given an fMDS $\mathfrak{m}_i : (\Sigma \rightarrow \mathsf{e})$, and a set of fMDSs $\mathfrak{M}$ where $\mathfrak{m}_i \in \mathfrak{M}$, the closure of $\mathfrak{m}_i$ is the function $\mathfrak{m}_i^+ : (\Sigma \rightarrow \mathcal{E})$ such that

$$\mathcal{E} = \Sigma \cup \{\mathsf{e}\} \cup (\bigcup \forall \mathfrak{m}_j^{+RHS} \{\mathfrak{m}_j \in \mathfrak{M} | \mathfrak{m}_j^{+LHS} \subseteq \mathfrak{m}_i^{+LHS}\})$$

where $\mathfrak{m}^{+LHS}$ denotes the signature Σ, and $\mathfrak{m}^{+RHS}$ refers to signature $\mathcal{E}$.

The closure of a set of fMDSs $\mathfrak{M}$ is the set $\mathfrak{M}^+$, where all fMDS $\mathfrak{m}_i^+ \in \mathfrak{M}^+$ is the closure of the corresponding $\mathfrak{m}_i \in \mathfrak{M}$. The following example illustrates fMDS closures:

Example 7.4 (fMDS set closure)**.** *Let $\mathfrak{M}$ be a set of fMDS, and $\mathfrak{M}^+$ the closure of $\mathfrak{M}$ such that*

- $\mathfrak{M} = \{\mathfrak{m}_1 : (\mathsf{A}, \mathsf{B} \rightarrow \mathsf{C}), \mathfrak{m}_2 : (\mathsf{B} \rightarrow \mathsf{D})\}$
- $\mathfrak{M}^+ = \{\mathfrak{m}_1^+ : (\mathsf{A}, \mathsf{B} \rightarrow \mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}), \mathfrak{m}_2^+ : (\mathsf{B} \rightarrow \mathsf{B}, \mathsf{D})\}$

The closure of fMDSs is computed as follows:

1. *signature $\mathfrak{m}_1^{+LHS}$ in addition to implicitly covering concept C, also explicitly covers concepts A, B, hence $\mathfrak{m}_1^{+RHS} = \mathfrak{m}_1^{RHS} \cup \{\mathsf{A}, \mathsf{B}\}$;*
2. *$\mathfrak{m}_1^+$ implicitly covers D as $\mathfrak{m}_2^{+LHS} \subseteq \mathfrak{m}_1^{+RHS}$ thus $\mathsf{D} \in \mathfrak{m}_1^{+LHS}$;*
3. *$\mathfrak{m}_2$ explicitly covers concepts B, hence $\mathfrak{m}_2^{+RHS} = \mathfrak{m}_2^{RHS} \cup \{\mathsf{B}\}$;*
4. *no more inference rules apply, thus the closure is complete.*

The cost and value calculation of an fMDS is formalised as follows:

Definition 7.8 (fMDS value and cost)**.** Given an fMDS $\mathfrak{m}$, an ontology $\mathcal{O}$, a task signature $\mathfrak{S}$, a cover set $\mathfrak{C}$, and $\mathfrak{M}$ the complete set of fMDSs in $\mathcal{O}$, where $\mathsf{Sig}(\mathfrak{m}), \mathfrak{S} \subseteq \mathcal{O}$ the value and cost of $\mathfrak{m}$ with respect to $\mathfrak{S}$ and $\mathfrak{C}$ is given by

- the value function $v(\mathfrak{m})$, which assigns a natural number $i \in \mathbb{N}_0$ to $\mathfrak{m}$ such that $v(\mathfrak{m}) = |\mathfrak{R} \setminus \mathfrak{C}^+ \cap \mathfrak{m}^{+RHS}|$
- the cost function $c(\mathfrak{m})$, which assigns a natural number $i \in \mathbb{N}_0$ to $\mathfrak{m}$ such that $c(\mathfrak{m}) = |\mathfrak{C}^+ \setminus \mathfrak{m}^{LHS}|$

7.3.1 Computing Minimal Covers

Algorithm 15 present the *Greedy #2* approach that efficiently finds an approximation of the minimal cover set for an ontology signature. The algorithm employs two subroutines (presented in Section 7.3.2): Algorithm 17, which computes the *closure of signatures*, and Algorithm 16 that provides the *closure of a set fMDSs*.

Algorithm 15: ComputeMinimalSignatureCover($\mathcal{O}, \mathfrak{R}, \mathfrak{S}, \mathfrak{M}$)

Input : $\mathcal{O}$: ontology; $\mathfrak{S}$: task signature; $\mathfrak{R}$: restricted signature; $\mathfrak{M}$: the complete set of fMDSs of each definable entity $\mathsf{e} \in \mathcal{O}$

Output: $\mathfrak{C}$: cover set of $\mathfrak{S}$ w.r.t. $\mathfrak{R}$

1 $\mathcal{C} \leftarrow \mathfrak{C} \cup \forall \mathsf{e}\{\mathsf{e} \in \mathfrak{S} \cap \mathfrak{R} | \mathsf{e} \notin \mathfrak{m}_i^{RHS} | \mathfrak{m}_i \in \mathfrak{M}\}$
2 $\mathfrak{M}^+ \leftarrow \mathsf{Initalise}(\mathfrak{M})$
3 $\mathfrak{C}^+ \leftarrow$ ComputeSignatureClosure($\mathfrak{C}, \mathfrak{M}^+$)
4 $\overline{\mathfrak{M}^+} \leftarrow \mathfrak{M}^+ \setminus \forall \mathfrak{m}_i\{\mathfrak{m}_i \in \mathfrak{M}^+ | \mathfrak{m}_i^{LHS} \subseteq \mathfrak{C}^+\}$
5 **while** $(\mathfrak{S} \setminus \mathfrak{C}^+) \neq \emptyset$ **do**
6 $\quad \mathcal{V} \leftarrow$ ComputeValueCostVector($\mathfrak{M}^+, \mathfrak{S}, \mathfrak{C}^+$)
7 $\quad \mathfrak{m}_{selected} \leftarrow$ select an $\mathfrak{m} \in \overline{\mathfrak{M}^+}$ according to $\mathcal{V}$, with max $value(\mathfrak{m})$, and min $cost(\mathfrak{m})$
8 $\quad \mathfrak{C} \leftarrow \mathfrak{C} \cup \mathfrak{m}_{selected}^{LHS}$
9 $\quad \mathfrak{C}^+ \leftarrow$ ComputeSignatureClosure($\mathfrak{C}, M$)
10 $\quad \overline{\mathfrak{M}^+} \leftarrow \overline{\mathfrak{M}^+} \setminus (\{\mathfrak{m}_{selected}\} \cup \forall \mathfrak{m}_i\{\mathfrak{m}_i \in \overline{\mathfrak{M}^+} | \mathfrak{m}_i^{LHS} \subseteq \mathfrak{C}^+\})$
11 **end**
12 **return** $\mathfrak{C}$

Walkthrough. Algorithm 15 assumes the *precondition*, that the task signature $\mathfrak{S}$ is coverable by the restricted signature $\mathfrak{R}$ (i.e. $\mathfrak{S} \subseteq \mathfrak{R}^+$). The algorithm starts by applying an *optimisation heuristic*, which reduces the search space; the heuristic initialises the cover set $\mathfrak{C}$ with all entities of $\mathfrak{S}$ that can only be covered explicitly by $\mathfrak{R}$ (line 1). Next $\mathfrak{M}$, which is used as the search space, is initialised with the complete set of fMDSs $\mathfrak{M}$. In addition, the process generates an 'aritfical' MDS and stores it in $\mathfrak{M}^+$, for each entity that can be covered both explicitly and implicitly. For instance, given a concept A the generated fMDSs is $\mathfrak{m}$: $(\mathsf{A} \rightarrow \mathsf{A})$, i.e. the entity can cover itself as it is permitted by the restricted signature (i.e. $\mathsf{A} \in \mathfrak{R}$). By including such artificial fMDSs that do not originate from actual minimal definition signatures, the algorithm ensures that the search space is complete, i.e. for each task signature entity the search space $\mathfrak{M}^+$ includes all possible ways of cover. The initialisation is concluded by computing the fMDS set closure (line 2).

Before the process begins the search, $\mathfrak{C}^+$, the cover closure is computed. This facilitates the *termination condition of the search* process (line 5), which halts the algorithm when task signature is covered ($\mathfrak{S} \setminus \mathfrak{C}^+ = \emptyset$). Then $\overline{\mathfrak{M}^+}$ is created as a copy of $\mathfrak{M}^+$, the former is the actual search space which is continuously pruned at each iteration (in order to optimise the process, by reducing the size of the search space and subsequently the effort required to calculate the cost and value scores of fMDSs), while the latter is the complete set of fMDSs closures which is left intact for the purpose of computing

signature closures during the search. The pruning of $\overline{\mathfrak{M}^+}$ is carried out by removing any fMDSs whose value (and cost) w.r.t. the cover set is zero (line 4), i.e. the fMDSs whose LHS signature already appears in $\mathfrak{C}$.

During the search (line 5-11), the value and cost of each fMDSs in $\mathfrak{m}_i \in \overline{\mathfrak{M}^+}$ is evaluated w.r.t. the cover set (line 6), then the best fMDSs is selected (line 7) and the LHS of the fMDS, which is the minimal definition signature, is added to the cover (line 8). The cover is then reevaluated by updating its closure (line 9), finally $\overline{\mathfrak{M}^+}$ is pruned according to the updated cover set. These steps are repeated until the task signature is covered, at which point the algorithm returns $\mathfrak{C}$, a non-redundant set that covers the task signature.

Correctness. The algorithm always finds a non-redundant cover set for a given task signature, which is an approximation of the minimal cover set. Non-redundancy is ensured by the selection function, and the fact that the entities added to the cover are MDSs, i.e. already minimal entity sets that are required to cover an other entity.

Termination and Complexity. At the worst case, the process covers at least one entity at each iteration, thus the maximum number of steps performed by the algorithm is n, where $n = |\mathfrak{S}|$. As both subroutines (Algorithm 17 and 16) employed by this algorithm have polynomial time computational complexity, it holds that the overall complexity of this algorithm is polynomial in the size of the input as well (please note that the algorithm assumes that the MDSs are precomputed). Moreover, as both of subroutines terminate and the halting condition (line 5) suspends the main loop of Algorithm 15 when the cover set is complete, it follows that the algorithm terminates.

7.3.2 Computing Signature and MDS Closures

fMDS Closure. Algorithm 16 computes the closure of an fMDS set (where the closure of each fMDS consists of all entities that can be covered using the LHS signature of an fMDS, i.e. the MDS) by exhaustively applying the inference rules given by Definition 7.7 and illustrated by Example 7.3. The process first applies the rule, that each entity is explicitly definable by itself, to every fMDS $\mathfrak{m} \in \mathfrak{M}^+$ (line 3). Next the algorithm enters a loop, which terminates when no further update is possible, i.e. the closure is complete for all fMDSs; during this process each fMDS is iteratively updated by adding all implicitly definable entities to the RHS of any fMDS whose LHS is and MDS (line 11). At the worst case, the each fMDS is 'applied' to every other fMDS, at most once, thus the algorithm performs approximately (less than) in quadratic time.

Signature Closure. Algorithm 17 computes the closure of an ontology signature w.r.t. to a set of fMDSs, in a similarly fashion as Algorithm 16, by iteratively applying the implicit definability relation given by the fMDSs. $\mathfrak{X}^+$, the closure of a signature consists of all entities that can be covered either explicitly or implicitly by

Algorithm 16: ComputeMDSClosure($\mathfrak{M}$)

Input : $\mathfrak{M}$: the complete set of fMDSs of each definable entity $\mathsf{e} \in \mathcal{O}$
Output: $\mathfrak{M}^+$: fMDSs closure

$\mathfrak{M}^+ \leftarrow \mathfrak{M}$
for $\mathfrak{m} \in \mathfrak{M}^+$ **do**
  $\mathfrak{M}^{RHS} \leftarrow (\mathfrak{m}^{LHS} \cup \mathfrak{m}^{RHS})$
end
Updated $\leftarrow$ true
while Updated **do**
  Updated $\leftarrow$ false
  for $\mathfrak{m} \in \mathfrak{M}^+$ **do**
    for $\mathfrak{m}' \in \mathfrak{M}^+$ **do**
      if $\mathfrak{m}' \neq \mathfrak{m} \wedge \mathfrak{m}^{LHS} \subseteq \mathfrak{m}'^{RHS} \ \wedge \ \mathfrak{m}^{RHS} \not\subseteq \mathfrak{m}'^{RHS}$ **then**
        $\mathfrak{m}' := (\mathfrak{m}'^{LHS} \rightarrow \mathfrak{m}'^{RHS} \cup \mathfrak{m}^{LHS})$
        Updated $\leftarrow$ true
      end
    end
  end
end
return $\mathfrak{M}^+$

Algorithm 17: ComputeSignatureClosure($\mathfrak{X}, \mathfrak{M}^+$)

Input : $\mathfrak{X}$: entity set of an ontology $\mathcal{O}$;
  $\mathfrak{M}$: the complete set of fMDSs of each definable entity $\mathsf{e} \in \mathfrak{X}$ under $\mathcal{O}$
Output: $\mathfrak{X}^+$: $\mathfrak{X}$ closure

$\mathfrak{X}^+ \leftarrow \mathfrak{X}$
Updated $\leftarrow$ true
while Updated **do**
  Updated $\leftarrow$ false
  for $\mathfrak{m}^+ \in \mathfrak{M}^+$ **do**
    if $\mathfrak{m}^{+LHS} \subseteq \mathfrak{X}^+$ **then**
      $\mathfrak{X}^+ \leftarrow \mathfrak{X}^+ \cup \mathfrak{m}^{+RHS}$
      $\mathfrak{M}^+ \leftarrow \mathfrak{M}^+ \setminus \{\mathfrak{m}^+\}$
      Updated $\leftarrow$ true
    end
  end
end
return $\mathfrak{X}^+$

the set $\mathfrak{X}$. For example, given a signature $\mathfrak{X} = \{\mathsf{A}, \mathsf{B}\}$, and the set of fMDSs closures $\mathfrak{M}^+ = \{\mathfrak{m}_1^+ = (\mathsf{A}, \mathsf{B} \rightarrow \mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}), \mathfrak{m}_2^+ = (\mathsf{C} \rightarrow \mathsf{D})\}$ the closure of $\mathfrak{X}$ is given by the set $\mathfrak{X}^+ = \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}\}$. The algorithm has polynomial worst case time complexity, as at most every fMDS is applied to the signature exactly once.

Supporting Incomplete MDSs. While both closure computation algorithms are implemented as loops, this is not necessary when the complete set of MDSs is available.

However, when the MDS set is incomplete, closure algorithms effectively perform the same notion as Algorithm 14, which expands existing MDSs by rewriting definable entities in MDSs.

7.3.3 Filtering Redundant Covers

Example 7.3 have outlined an algorithm (*Greedy #1*), which produces redundant cover sets. The main difference between *Greedy #1* and #2, and the cause of the non-redundancy, is that the former narrows the non-deterministic part of the search, as it explicitly covers all of those entities that can be covered explicitly in addition to being implicitly coverable as well. This is achieved by the initialisation heuristic:

$$\mathfrak{C} \leftarrow \mathfrak{C} \ \cup \ \forall \mathsf{e}\{\mathsf{e} \in (\mathfrak{S} \cap \mathfrak{R})\} \tag{7.1}$$

Thus approach #1 is typically faster than #2, however #2 may produces a considerably more optimal cover set compared to approach #1. The redundancy issue of approach #1 is trivial to resolve, as redundant entities can be filtered out from the initially non-redundant solution, in polynomial time, using Algorithm 18 which iteratively removes every such entity from the cover set $\mathfrak{C}$ that is otherwise implicitly coverable by $\mathfrak{C}$. Thus the approach *Greedy #3* which extends #1 by performing the aforementioned filtering, is expected to produce more optimal solutions than #1, while typically performing faster than approach #2. *Greedy #3* is implemented by simply modifying the initialisation heuristic (line 1) of Algorithm 15, as given by (7.1); moreover applying the filtering once the search phase concluded.

Algorithm 18: FilterCoverSet($\mathfrak{C}, \mathfrak{M}^+$)

Input : $\mathfrak{C}$: potentially redundant cover set;
$\mathfrak{M}^+$: fMDSs set closure
Output: $\mathfrak{C}$: non-redundant cover set

```
1  C+ ← ComputeSignatureClosure(C, M+)
2  while True do
3  |  for e ∈ C do
4  |  |  if ∃m{m ∈ M|e ∈ m^RHS|m^LHS ⊆ C} then
5  |  |  |  C ← C \ {e}
6  |  |  |  break
7  |  |  end
8  |  |  return C
9  |  end
10 |  C+ ← ComputeSignatureClosure(C, M+)
11 end
```

Algorithm 18 filters a given, potentially redundant, cover set by removing all entities that are both implicitly and explicitly definable. All removable entities cannot be selected and filtered out from $\mathfrak{C}$ at the same time as any removed entity may be required for an other MDS, thus the process iteratively identifies removable entities by

assessing the cover set closure, and removing one entity at the time. The process runs in polynomial time and terminates when $\mathfrak{C}$ contains no superfluous entities.

7.3.4 Computing Multiple Approximations

A given restricted signature can contain more than one non-redundant cover set, where some covers are potentially pairwise disjoint. Algorithm 19 presents an approach, which identifies a set of disjoint covers, by employing either the *Greedy #2* or *#3* as sub-routine to compute a single approximation.

Algorithm 19: ComputeMultipleMinimalSignatureCovers($\mathcal{O}, \mathfrak{R}, \mathfrak{S}, \mathfrak{M}$)

Input : $\mathcal{O}$: ontology; $\mathfrak{S}$: task signature; $\mathfrak{R}$: restricted signature;
$\mathfrak{M}$: the complete set of fMDSs of each definable entity $\mathsf{e} \in \mathcal{O}$
Output: $\mathfrak{K}$: set of (disjoint) approximated minimal cover set of $\mathfrak{S}$ w.r.t. $\mathfrak{R}$

1 $\mathfrak{M}^+ \leftarrow \mathsf{Initalise}(\mathfrak{M})$
2 $\overline{\mathfrak{R}} \leftarrow \mathfrak{R}$
3 $\overline{\mathfrak{R}^+} \leftarrow$ ComputeSignatureClosure($\overline{\mathfrak{R}}, M^+$)
4 **while** $\mathfrak{S} \subseteq \overline{\mathfrak{R}+}$ **do**
5 $\quad \mathfrak{C} \leftarrow$ ComputeMinimalSignatureCover($\mathcal{O}, \mathfrak{R}, \mathfrak{S}, \mathfrak{M}$)
6 $\quad \mathfrak{K} \leftarrow \mathfrak{K} \cup \{\mathfrak{C}\}$
7 $\quad \overline{\mathfrak{R}} \setminus \mathcal{C}$
8 $\quad \overline{\mathfrak{R}^+} \leftarrow$ ComputeSignatureClosure($\overline{\mathfrak{R}}, M^+$)
9 **end**
10 **return** $\mathfrak{K}$

The algorithm first initialises the fMDS closure $\mathfrak{M}^+$ (line 1), the 'working' restricted signature $\overline{\mathfrak{R}}$ (line 2), and the closure of the restricted signature $\overline{\mathfrak{R}^+}$ (line 3). Next the process enters a loop, where at each iteration the algorithm computes a cover set $\mathfrak{C}$ (line 5), and removes its members from $\overline{\mathfrak{R}}$ (line 7). The process terminates when all disjoint covers have been identified, i.e. $\mathfrak{S} \not\subseteq \overline{\mathfrak{R}+}$.

The algorithm runs in polynomial time, as at the worst case the number of disjoint covers sets is equivalent to the restricted signature cardinality, moreover, both subroutines used by the algorithm are polynomial as well. Termination is ensured by the loop condition, which is halted when the iteratively pruned restricted signature not longer covers the task signature.

7.3.5 Computing Full Covers

Computing coverage for the entire TBox signature (i.e. $\mathfrak{S} = \mathsf{Sig}(\mathcal{T})$) is a special case because for such task signature it is possible to efficiently find either a very close approximation, or even the actual minimal signature cover set.

A *full cover* excludes all definable concepts as these are implicitly coverable using undefined concepts and (both definable and undefined) roles; and includes all undefined entities, along with all individual names that appear in $\mathsf{Sig}(\mathcal{T})$, that can only be covered explicitly. Therefore, if the ontology contains no roles, the actual minimal cover can

Algorithm 20: ComputeFullTBoxCover($\mathsf{e}, \mathcal{T}, \mathfrak{M}$)

Input : $\mathfrak{S}$: task signature, such that $\mathfrak{S} = \mathsf{Sig}(\mathcal{T})$; $\mathcal{T}$: TBox;
$\mathfrak{M}$: fMDSs of all $\mathsf{e} \in \mathfrak{S}$
Output: $\mathfrak{C}$: cover set of $\mathfrak{S}$

```
1 𝔠 ← Sig(𝒯)
2 𝔠 ← 𝔠 \ ∀e{∃𝔪|𝔪 ∈ 𝔐|e ∈ 𝔪^RHS}
3 𝔠 ← FilterCoverSet(𝔠, 𝔐⁺)
4 return 𝔠
```

be obtained by simply including all undefined concepts. However, covering roles is not as trivial: first all roles are added to the signature, then any redundant, (definable) roles are filtered out from the cover, using Algorithm 18. Considering redundancy is necessary in order to avoid removing those roles that are both definable, and provide the only definition signature to another entity, for example, the axiom $\mathsf{r} \equiv \mathsf{s}$ implies that both roles r and s are definable, however removing both entities from the cover set would leave them both uncovered.

Algorithm 20 implements this process. As the filtering subroutine runs in polynomial time, the computational time complexity of computing full covers is polynomial as well.

7.4 Empirical Evaluation

The aim of the evaluation was to empirically determine how effective the presented approximation approaches are in finding task signature cover sets. The experiments tested the *hypothesis*, that the presented approximation approaches, by employing both explicit and implicit coverage, produce a cover set which is albeit not minimal, but still considerably smaller than cover sets obtained by only explicit coverage. Thus approximations of minimal cover sets are typically smaller than explicit covers, if the given task signature contains definable entities w.r.t. a restricted signature (clearly, for a task signature which lacks definable entities, only explicit coverage is possible that are by default minimal cover sets). In addition, by measuring the computation *time* and the *cardinality* of the resulting cover sets, the evaluation compares the *Greedy #2* and *#3* approaches, that produce more optimal approximations, i.e. non-redundant, approximated cover sets. Greedy #2 considers a larger search space, thus it is expected to provide a more optimal solution (i.e. a smaller cover set) than Greedy #3, consequently, the latter approach is likely to perform faster due to the smaller search space.

The first experiment (Section 7.4.1) provides insight into the effectiveness of the approximation. An obvious easy case when the actual minimal cover set is efficiently computable occurs when the task signature is equivalent to the ontology signature, thus this provides the opportunity to compare the result of the 'ideal cover set', and the approximation produced by *Greedy #2* and *#3*. The second experiment (Section 7.4.2) employs a range of different task signature sizes to assess the reduction provided by a minimal cover in comparison to the baseline (explicit cover), and to evaluate the size

Ontology	$\lvert N_{\mathcal{C}} \cup N_{\mathcal{R}} \rvert$	$Def\%$	*Ideal Cover*	*Greedy #3*		*Greedy #2*	
			cov	*cov*	Time	*cov*	Time
Conference corpus							
cmt	88	50.00%	50.00%	72.73%	0.48 ms	72.73%	3.62 ms
conference	123	57.72%	42.28%	69.92%	2.54 ms	70.73%	11.79 ms
confOf	74	12.16%	87.84%	89.19%	0.25 ms	89.19%	0.43 ms
edas	153	26.14%	73.86%	85.62%	3.05 ms	86.27%	8.70 ms
ekaw	106	28.30%	71.70%	85.85%	0.20 ms	85.85%	2.30 ms
iasted	181	17.68%	82.32%	87.85%	0.96 ms	87.85%	3.18 ms
sigkdd	77	25.97%	74.03%	81.82%	0.43 ms	81.82%	1.12 ms
AVG.	**114.57**	**31.14%**	**68.86%**	**81.85%**	**1.13 ms**	**82.06%**	**4.45 ms**
LargeBio corpus							
NCI_fma	6551	29.98%	70.02%	81.30%	1.53 s	70.02%	3.09 s
NCI_snomed	24040	28.50%	71.50%	82.61%	27.68 s	71.50%	56.42 s
SNOMED_fma	13430	21.47%	78.54%	85.82%	4.36 s	78.56%	8.64 s
SNOMED_nci	51128	57.87%	42.13%	62.18%	709.76 s	42.45%	838.30 s
AVG.	**23787.25**	**34.46%**	**65.55%**	**77.98%**	**185.83 s**	**65.63%**	**226.61 s**

TABLE 7.3: Comparing cover size and computation of time of approach #2 and #3, for full covering the entire ontology signature.

and the computation time difference between the two approaches, on a wider scale of possible tasks size settings.

Evaluation Corpus. The *evaluation corpus* consists of 7 small ontologies (average 70.14 concepts and roles per ontology) from the *Conference* dataset; and 4 large (average 23787.25 entities) ontologies from the *Large biomedical ontology* dataset. Thus the corpus is diverse in size, moreover, it is appropriate to assess implicit coverage as all ontologies contain some definable entities. For every concept and role in each ontology of the evaluation corpus, the definability status and the complete set of MDSs have been pre-computed.

Table 7.3 presents a summary of the corpus, showing number of concepts and roles in a given ontology signature ($|N_{\mathcal{C}} \cup N_{\mathcal{R}}|$), the ratio of definable concepts and roles ($Def\%$). Both datasets contain ontologies with varying level of definability, as shown by the ratio of definable ontology signature entities.

Experimental Framework and Conduct. The *experimental framework* was implemented in Java; entity definability status, and corresponding MDSs were computed using the OntoDef API which facilitated MDS finding experiments (Chapter 5.2).

In all experiments, for each task signature, cover sets were computed by using the two approximation approaches, *Greedy #2* and *#3*. A naive approach, which considers only explicit coverage of signatures and always provides a cover that is the same set as the task signature (i.e. it is a constant $cov = 100\%$) was used as the *baseline* in all experiments. The evaluation have only considered *coverable* task signatures (i.e. $\mathfrak{S} \subseteq \mathfrak{R}^{+}$), thus in all cases, the restricted signature $\mathfrak{R}$ was equivalent to the T-Box signature, while the task signature $\mathfrak{S}$ was allocated several differently sized T-Box signature subsets (i.e. $\mathfrak{R} = \mathsf{Sig}(\mathcal{T})$, and $\mathfrak{S} \subseteq \mathsf{Sig}(\mathcal{T})$). Varying the composition of only one of the two

signatures simplified both the experiment conduct and the result analysis process, while it provided the same overall results. Experiments were conducted with 8GB maximum memory allocated for the Java Virtual Machine, running on a machine equipped with 16GB RAM and a 16 core processor architecture.

7.4.1 Experiment 1: Ideal Covers

This experiment compares the cover set obtained by the different approximation approaches, to the actual minimal cover set. An easy case of the minimal signature cover, which is not an approximation and can be computed efficiently is when the task requires the entire signature of an ontology to be covered, i.e. $\mathfrak{S} = \mathsf{Sig}(\mathcal{O})$. This special case provides the opportunity to evaluate the difference between an actual, and an approximated minimal signature cover set.

An *ideal cover* is obtained by removing all non-redundant, definable concept and role names from the ontology signature, using Algorithm 18. Considering non-redundancy in entity removal is necessary in order to avoid removing those entities that are both definable, and provide the only definition signature to another entity, for example, the axiom $\mathsf{r} \equiv \mathsf{s}$ implies that both roles r and s are definable, however removing both entities from the ideal cover would make them both uncoverable.

Results are presented in Table 7.3 (where the baseline method is omitted). The partition labeled ideal cover shows the size of the minimal cover set in ratio to the an explicit cover, which is always equivalent to the task signature (i.e. $cov = \frac{|\mathfrak{C}|}{|\mathfrak{S}|}$), while the two right hand side partitions present the result obtained with the greedy algorithms, in terms of the cover size ratio, and the computation time. In the Conference corpus, which contains small ontologies, neither approaches have come close to the ideal cover set (which was, on average 68.86%). The cover sets produced by greedy #2 and #3, on average were 81.85% and 82.06% of the task signature, respectively. With the exception of two cases (small ontologies *conference* and *edas*), where #2 produced slightly larger cover sets than #3 (the difference in their average is 0.21%), the two algorithms have produced the same results. In the LargeBio corpus, on average, the size of the minimal cover set was 65.55% of task signature, thus with a 65.63% average, approach #2 performed significantly better than #2 (77.98%), and with only a 0.08% difference, it has nearly achieved the optimal solution in all large ontologies, i.e. the minimal cover. In terms of computation time, as expected, greedy #3 performed considerably faster in both datasets: in the Conference corpus, on average, greedy #3 has took 1.13 milliseconds to compute a cover set, while greedy #2 has completed the same task in 4.45 milliseconds; in the LargeBio corpus, greedy #3 needed 185.83 seconds, while #3 required 226.61 seconds to complete the search process.

7.4.2 Experiment 2: Varying signatures

The second experiment varied the size of the task signature size, in order to assess the reduction provided by a minimal cover in comparison to the baseline (explicit cover), and

to evaluate the size and the computation time difference between the two approaches, on a wider scale of possible tasks size settings. This experiment included 20 test cases for each ontology, where the task to ontology signature ratio ranged between 100% and 5%. Due to the fact that both approaches include an non-deterministic part, where a random choice is made to select an MDS from a set of equally good options (MDSs with the same value and cost scores), each test case was repeated 100 times (several different repetition counts were tested, comparing their relative standard deviation established that 100 repetition was sufficient for both datasets).

Figure 7.1 presents the *cover set cardinality* results, where the y-axis represents the approximated minimal cover to task signature ratio ($\frac{|\mathfrak{C}|}{|\mathfrak{S}|}$), and the x-axis shows the task to ontology signature ratio ($\frac{|\mathfrak{S}|}{|\mathsf{Sig}(\mathcal{O})|}$). Figure 7.1 shows the results of the Conference dataset, computed by approach #2 (a), and approach #3 (b); and the results of the LargeBio dataset for approach #2 (a) and #3 (b), respectively (for brevity, error bars are only shown for the ontologies with the highest and lowest covers). In the small ontologies of the Conference corpus, approach #3 performed slightly better than #2, with all task signature sizes, while in LargeBio corpus, approach #2 performs considerably better for larger task signatures, however, with smaller task signatures (under 40%) approach #3 still provides better results. This is reinforced by the data presented in Figure 7.1 (e) and (f), which shows the cover cardinality results for the same ontologies, produced by the two approaches (in the *conference*, and *NCI_fma* ontologies from the Conference, and the LargeBio corpus, respectively).

Figure 7.2 present the *computation time* results, where the y-axis shows the time, and the x-axis shows the task to ontology signature size ratio. Figure 7.2 shows the results of approach #2 (a) and #3 (b) in the Conference corpus, while (c) and (d) provides the result of the two approaches in the LargeBio corpus, respectively. The computation time of approach #3 is almost a linear for a given ontology, while approach #2 seems to correspond to a bell curve (this is more visible in larger ontologies, as shown by 7.2, c).

In all ontologies of both datasets, approach #2 is significantly slower than #3, for example in the *NCI_fma* ontology (Figure 7.2, d), at 50% task to ontology signature ratio, #3 required 260.03 seconds, while #2 took only 1.59 seconds to complete. The same trend can be observed in the *conference* ontology (Figure 7.2, e).

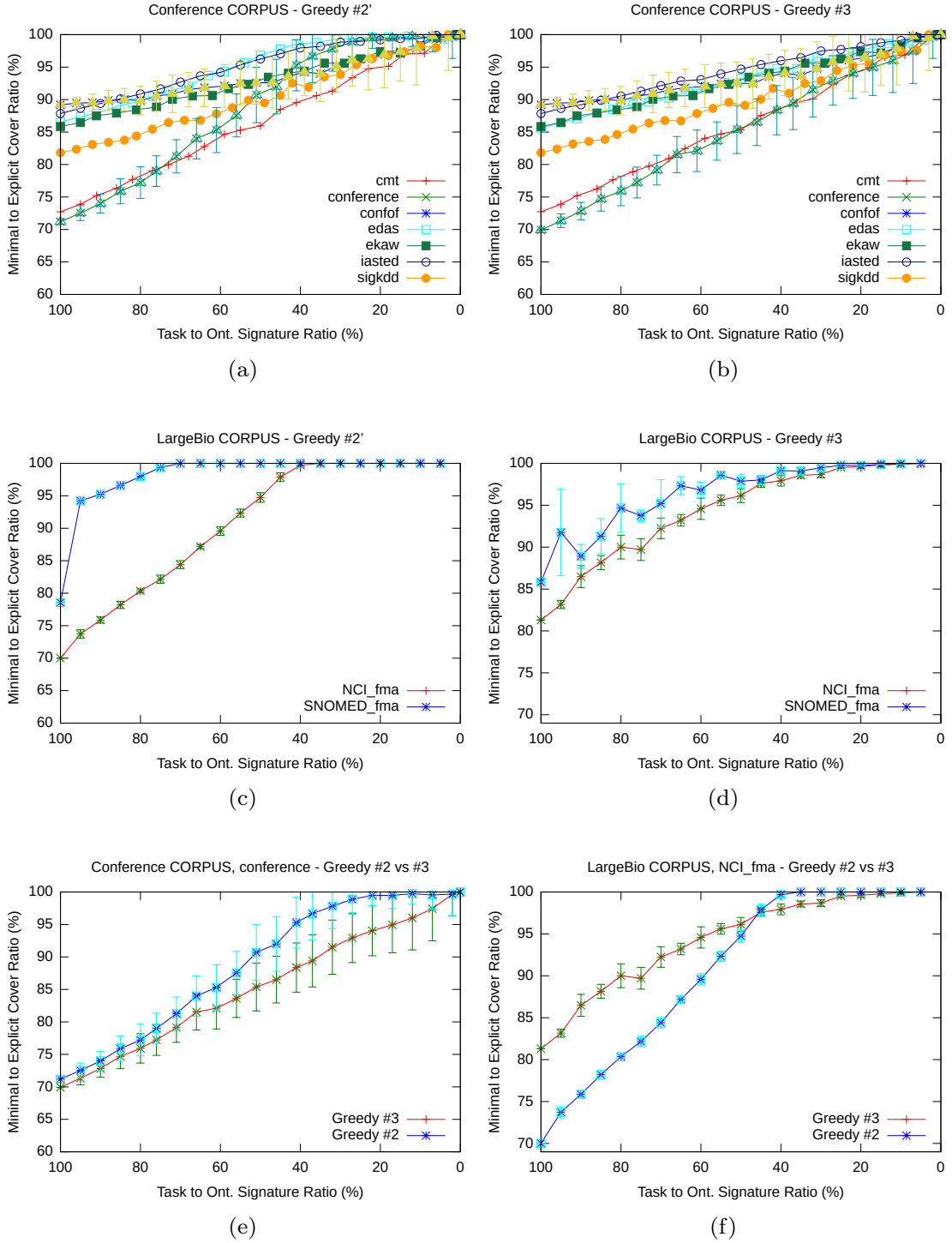


FIGURE 7.1: Cover set sizes, in the Conference (*a, b*) and LargeBio (*c, d*) corpus, obtained by the Greedy #2 (*a, b*) and #3 (*b, d*) approaches. Greedy #2 and #3 are compared in a Conference (*e*), a LargeBio ontology (*f*).

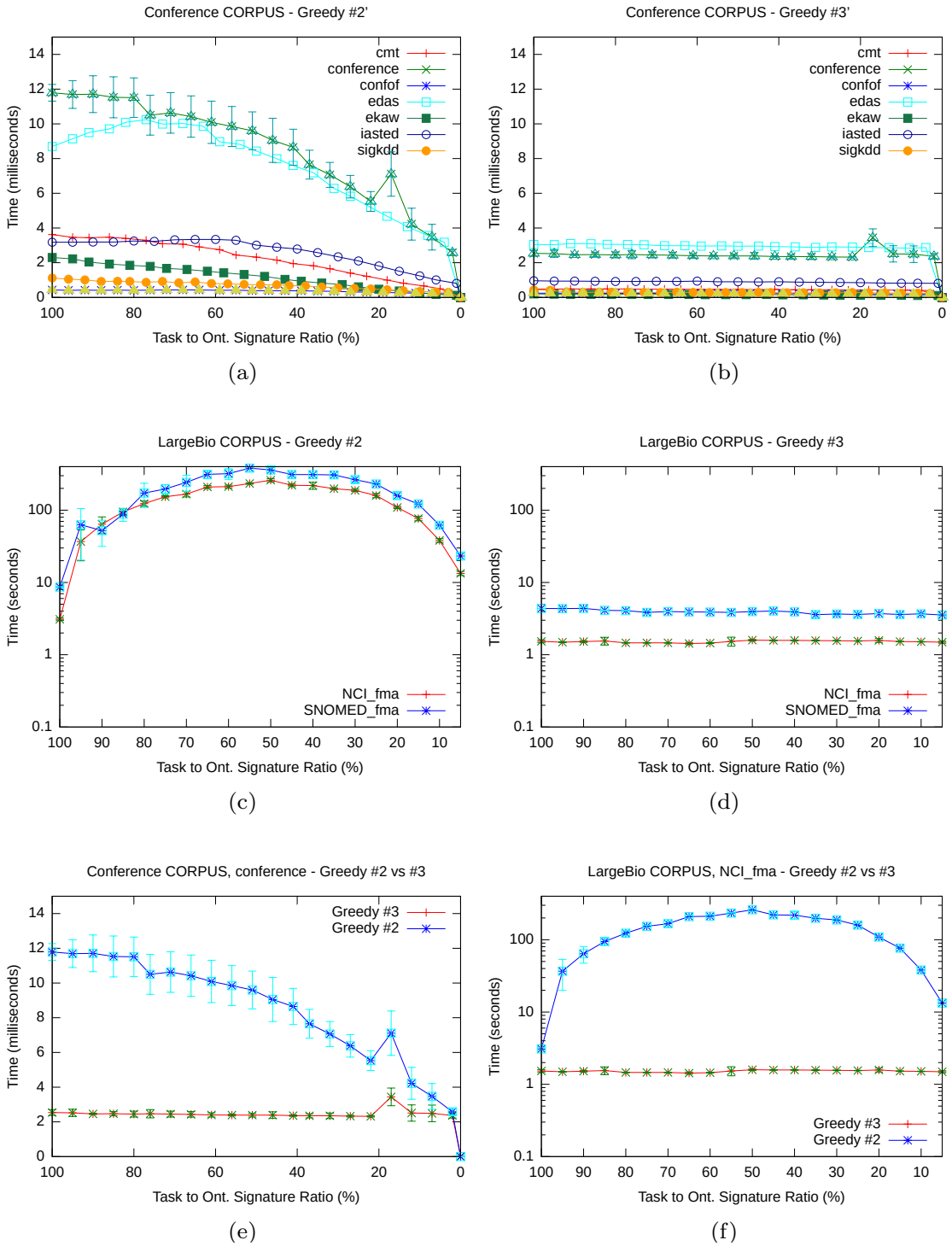


FIGURE 7.2: Cover set computation times, in the Conference (*a, b*) and LargeBio (*c, d*) corpus, obtained by the Greedy #2 (*a, b*) and #3 (*b, d*) approaches. Greedy #2 and #3 are compared in a Conference (*e*), a LargeBio ontology (*f*).

7.5 Summary and Conclusions

- This chapter have explored the notion of ontology signature coverage, which entails identifying whether a given ontology signature is covered with respect to another signature, and introduced the MDS-based, implicit signature coverage that potentially expands coverage to otherwise uncoverable signature entities.

- Whilst explicit coverage leaves no choice in composing cover sets as each entity is required to be represented directly, the union of explicit and implicit coverage permits different equally correct covers. Thus the chapter have introduced and characterised the non-polynomial, minimal ontology signature coverage problem, which concerns identifying the smallest cover set.

- Furthermore, we have presented and empirically evaluated two versions of the approximation approach, that by exploiting the notion of implicit definability and using the pre-computed, complete set of different MDSs, provides a sub-optimal solution to the minimal signature cover problem.

- The evaluation suggests that, although the resulting covers are not always optimal, i.e non-minimal, they are significant improvements on the covers obtained by only explicit coverage.

Chapter 8

Definability-based Correspondences and Alignment Evaluation

This chapter explores the application of (minimal) definition signatures in the context of ontology alignment. As implicit definability permits defined entities to be removed without semantic loss, this thesis argues, that if the meaning of the defined entity is wholly fixed by the terms of its definition, only the terms in the definition are required to be mapped in order to map the defined entity itself; thus implicit definability entails a new type of correspondence, *definability-based (or implicit) correspondence*, based on the definition signatures of entities in the aligned ontologies and the available alignment. Each correspondence describes a particular relation between a definition signature entity and a semantically related entity of the pairwise aligned ontology, thus a definability-based correspondences can only be considered if the semantic relations are compatible.

Furthermore, the chapter studies the *implications of considering definability-based correspondences in alignments*. An alignment provides a restricted signature over an ontology signature, thus the notion of implicit coverage also applies to alignments, which, as shown in Chapter 7 permits both extending the (alignment) coverage and reducing the size of the cover set. Considering implicit correspondences in alignments potentially increases alignment coverage, as such correspondences may map otherwise uncovered (but definable) entities; furthermore identifying a minimal sets of implicit correspondences, i.e. a minimal cover set under an alignment, may decrease alignment cardinality by reducing the number of required correspondences. Moreover alignments considering implicit correspondences can retain coverage at a better rate than traditional alignments, as entities can be mapped both directly and indirectly, hence removing some correspondences from the alignment does not effect its expressive power. Therefore alignments that consider implicit correspondences are expected to support semantic interoperability better than traditional ones.

One major issue, which hiders semantic interoperability, is that *alignments are typically incomplete* [47], providing only a partial coverage of an ontology signature, thus the use of implicit definability, by considering definability-based correspondences, can potentially fill in the resulting semantic gaps by expressing unmapped concepts or roles in terms of mapped entities, thus increase the utility of a given alignment. Moreover, knowing all different forms to express particular things (concepts, queries, etc.) is clearly advantageous in such scenarios where only a *restricted signature* is available. In order to be able to carry out meaningful communication, agents must cooperatively establish a mutually acceptable alignment, whilst adhering to internal preferences, without compromising confidential knowledge, and avoiding alignment-based conservativity violations to occur [83]. By considering definability-based evaluation metrics, agents are better equipped to determine whether an alignment provides the necessary coverage to achieve a particular task (align the whole ontology, formulate a message or query), as well as striving to minimalise the cardinality of the resulting mutual alignment.

Although several ontology alignment evaluation measures have been established to assess alignment quality [124], existing alignment evaluation metrics are insufficient for evaluating alignments with implicit correspondences, hence this chapter extends two evaluation metric families: (*i*) precision and recall that measure alignment quality by comparing produced alignments with a reference alignment; (*ii*) coverage and path-based metrics that assesses alignments quality and ontology proximity with respect to the coverage and distinguishability of entities, and to alignment paths (composed by sequences of alignments). These definability-based metrics support both the empirical evaluation of implicit correspondences, and are employed by knowledge-based agents in evaluating peers during coalition formation.

The remainder of this chapter is organised as follows: Section 8.1 formalises and characterises definability-based correspondences and discusses a possible issue: aligned ontologies might disagree on the definitions that form implicit correspondences. Section 8.2 extends the alignment evaluation metrics, that are computed in the alignment space, while Section 8.3.1 discusses the modified precision and recall metrics, which accounts for implicit correspondences. Section 8.4 reports on the empirical evaluation, including the experiment framework, methodology and the results. Section 8.5 summarises and concludes the chapter.

8.1 Definability-based Correspondences

Accounting for the notion of implicit definability in an alignment space results in new correspondences, by anchoring on simple correspondences and the axiomatisation of the aligned ontologies. The example represented by Figure 8.1 illustrates this notion; where three ontologies ($\mathcal{O}_1 - \mathcal{O}_3$) aligned by two alignments ($A_{1,2}$, $A_{2,3}$) form a knowledge-based network. The alignment $A_{1,2}$ connects two out of three named concepts of ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ but it provides no (direct) cover for the concept $\mathsf{A}_1 \in \mathcal{O}_1$ (or $\mathsf{A}_2 \in \mathcal{O}_2$). However,

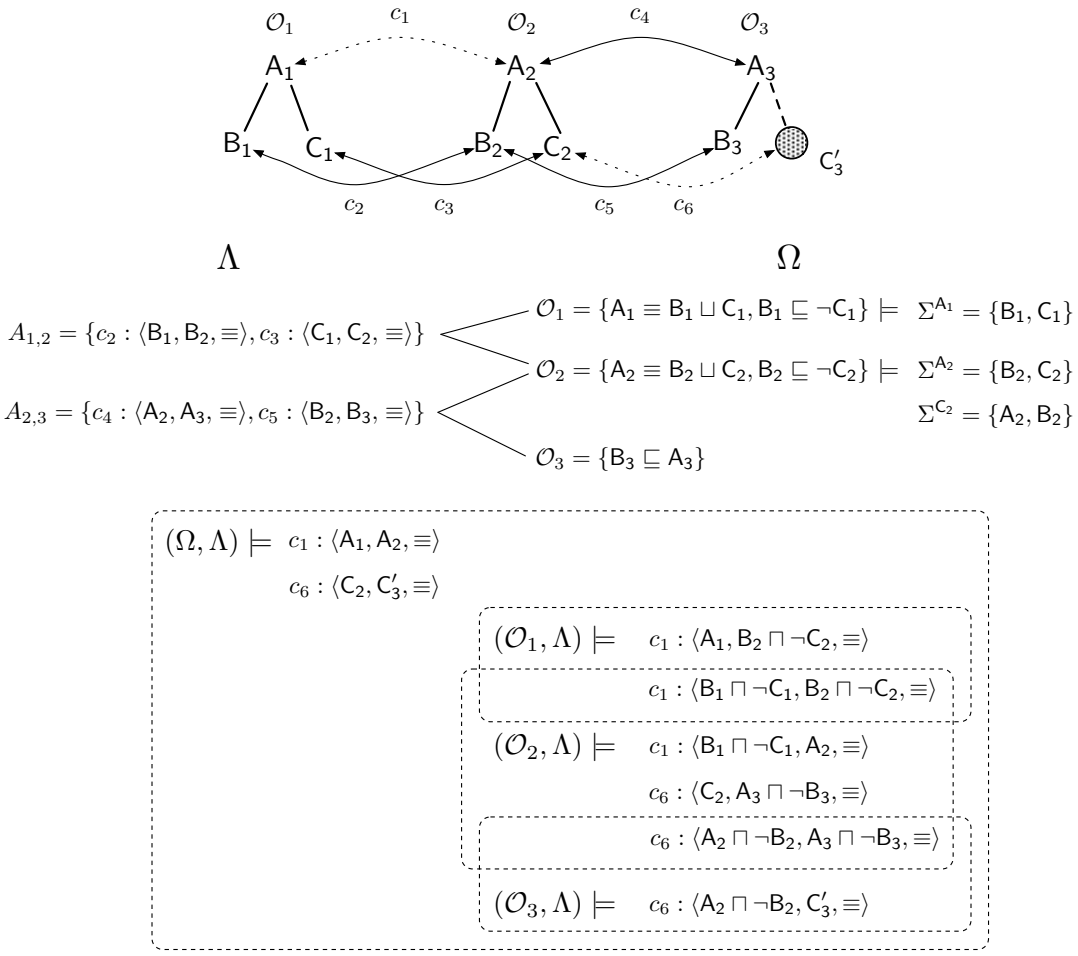


FIGURE 8.1: The alignment space contains explicitly $(\mathsf{B}_1, \mathsf{C}_1, \mathsf{A}_2, \mathsf{B}_2, \mathsf{C}_2, \mathsf{A}_3, \mathsf{B}_3)$, and implicitly mapped entities $(\mathsf{A}_1, \mathsf{A}_2, \mathsf{C}_2, \mathsf{C}'_3)$, where all entities are named, except the anonym concept C'_3. Simple correspondences $(c_2 - c_5)$ are represented as normal arcs, implicit correspondences (c_1, c_6) are denoted with dashed arcs.

as A_1 is implicitly definable under $\mathcal{O}_1$ by the MDS $\Sigma^{\mathsf{A}_1} = \{\mathsf{B}_1, \mathsf{C}_1\}$ which corresponds to a definition axiom $\mathsf{A}_1 \equiv \mathsf{C}_1 \sqcap \neg \mathsf{B}_1$, and all entities of Σ^{A_1} are mapped by equivalence correspondences (c_2, c_3), a new correspondence (c_1) can be created which establishes an equivalence relation between A_1 and A_2. Thus we can distinguish between an entity e being mapped *explicitly* by a correspondence that is asserted in an alignment (such as B_1 covered by c_2), or an *implicitly* mapped entity e' (e.g. A_1 mapped by c_1) which is entailed by a set of asserted correspondence that map all entities of a definition signature of e'. An explicitly mapped entity is defined as:

Definition 8.1 (Explicitly mapped entity)**.** An entity e is *explicitly mapped (or covered)* by an alignment A if and only if there exists a (potentially complex) correspondence $c \in A$ such that $c : \langle \mathsf{e}, \mathsf{e}', r\rangle$, where $\mathsf{e} \in \mathcal{O}$ and e' denotes either a named entity, or a complex concept or role of $\mathcal{O}'$.

Thus an entity can be mapped by an alignment explicitly, through either a simple, or a complex correspondence. An implicitly mapped entity is formalised as follows:

Definition 8.2 (Implicitly mapped entity)**.** Let us suppose that entity e_1 has a definition signature Σ^{e_1} in $\mathcal{O}_1$ and entity e_2 has a definition signature Σ^{e_2} in $\mathcal{O}_2$ and an alignment A maps the constituent entities of Σ^{e_1} to Σ^{e_2} where the correspondences in A either all describe the same relation, or their relations can be reconciled. Then e_1 and e_2 are implicitly mapped (or covered) by the alignment.

Clearly, only definable entities with valid definition signatures, i.e. concepts and roles (such that $\Sigma \neq \emptyset$), can be mapped implicitly. An correspondence that implicitly maps an entity, entailed by an entity definition and the available alignment, is referred to as an *implicit correspondence* and it is defined as follows:

Definition 8.3 (Implicit correspondence (or definability-based correspondence))**.** A definability-based correspondence c, which implicitly maps an entity $\mathsf{e} \in \mathcal{O}$ as entailed by the set of correspondences $A' \subseteq A$ of an alignment A between ontologies $\mathcal{O}$ and $\mathcal{O}'$, is a tuple $c : \langle \mathsf{e}, \mathsf{e}', r \rangle$, where e' either denotes an entity, a complex concept, or a complex role in $\mathcal{O}'$, and r is a relation between e and e' such that

- $r = \equiv$ iff $\forall c_i \{c_i \in A' | r_i \in \{\equiv\}\}$
- $r = \sqsubseteq$ iff $\exists c_i \{c_i \in A' | r_i \in \{\sqsubseteq\}\}$ and $\forall c_j \{c_j \in A' | r_j \in \{\sqsubseteq, \equiv\}\}$
- $r = \sqsupseteq$ iff $\exists c_i \{c_i \in A' | r_i \in \{\sqsupseteq\}\}$ and $\forall c_j \{c_j \in A' | r_j \in \{\sqsupseteq, \equiv\}\}$
- $r = \bot$ iff $\forall c_i \{c_i \in A' | r_i \in \{\bot\}\}$

Thus the definition prohibits *considering* a definability-based correspondence, if the relations of the supporting asserted correspondences are incompatible, for instance a set of relations $\{\sqsubseteq, \equiv\}$ is reconcilable to the weakest common relation $\sqsubseteq$, whereas the set $\{\bot, \equiv\}$ is incompatible.

Furthermore, a definability-based correspondence should not be considered if it violates the *conservativity principle* [83, 127]. This notion dictates, that the integrated ontology ($\mathcal{O} \cup \mathcal{O}' \cup A$) which is the product of merging a pairwise aligned ontology through an alignment, should not induce any change in the hierarchy of the input ontologies ($\mathcal{O}, \mathcal{O}'$). An implicit correspondence based on an erroneous alignment, or more precisely, the set of correspondences $A' \subseteq A$, may cause further logical flaws, thus it should not be considered. For example, let us consider the alignments $A_{0,1}$ and $A_{0,2}$ in Figure 8.4 (Section 8.2). In both cases an implicit correspondence that maps concept $\mathsf{C} \in \mathcal{O}_0$ could be created, as for alignments $A_{0,3}$ and $A_{0,4}$, because all entities of the definition signature of C are mapped. However, under $\mathcal{O}_0$, these entities are deemed disjoint, whereas under $\mathcal{O}_1$ and $\mathcal{O}_2$ these are mapped to the same concepts, thus contradict the concept hierarchy of $\mathcal{O}_1$.

Correspondence Formalisation. Depending on *perspective*, whether a *global view* ($\Lambda \cup \Omega$) of the alignment space is assumed which includes knowledge about all alignments and the internal axiomatisation of member ontologies, or a *local viewpoint* is of taken by a particular ontology in the alignment space ($\Lambda \cup \mathcal{O}_i$, where $\mathcal{O}_i \in \Omega$), an implicit correspondence can be formalised in several ways. For instance in Figure 8.1, c_1 may take one the following forms

- $c_1 : \langle \mathsf{A_1}, \mathsf{A_2}, \equiv \rangle$ under a global view, however, when taking the perspective of a particular ontology which lacks knowledge about the axiomatisation (thus the potential MDSs) of other ontologies;
- $c_1 : \langle \mathsf{A_1}, \mathsf{C_2} \sqcap \neg \mathsf{B_2}, \equiv \rangle$ under $(\Lambda \cup \mathcal{O}_1)$, if the concept name $\mathsf{A_1}$ is made public;
- $c_1 : \langle \mathsf{C_1} \sqcap \neg \mathsf{B_1}, \mathsf{C_2} \sqcap \neg \mathsf{B_2}, \equiv \rangle$ under $(\Lambda \cup \mathcal{O}_1)$ or $(\Lambda \cup \mathcal{O}_2)$ where neither defined, aligned concept names $(\mathsf{A_1}, \mathsf{A_2})$ are public;
- $c_1 : \langle \mathsf{C_1} \sqcap \neg \mathsf{B_1}, \mathsf{A_2}, \equiv \rangle$ under $(\Lambda \cup \mathcal{O}_2)$, if the concept name $\mathsf{A_2}$ is made public.

Aligned Entities. An entity of the target ontology, that is aligned to a defined entity of the source ontology by a definability-based complex correspondence can be classified either as a *named*, or an *anonym* (complex) concept or role. For example in Figure 8.1, regardless of the formalisation of the correspondence c_1, which aligns named concepts $\mathsf{A_1}$ and $\mathsf{A_2}$, the complex concepts appearing in the different instantiations of c_1 are named concept names under their respective ontologies. Ontology $\mathcal{O}_3$ has different granularity than $\mathcal{O}_2$, as it does not describe the concept C as the other ontologies, however c_6 establishes a relation between a named concept $\mathsf{C_2} \in \mathcal{O}_2$, and an anonym concept $\mathsf{C'_3} \in \mathcal{O}_3$, where $\mathsf{C'_3} \equiv \mathsf{A_2} \sqsubseteq \neg \mathsf{B_3}$. Thus the anonym concept $\mathsf{C'_3}$ has the same logical properties in $\mathcal{O}_3$ as if it was a named entity, i.e. $\mathcal{O}_2 \cup \mathcal{O}_3 \cup A_{2,3} \models \{\mathsf{C'_3} \sqsubseteq \mathsf{A_3}, \mathsf{C'_3} \sqcap \neg \mathsf{B_3}\}$.

Correspondence Classification. *An instance of a definability-based correspondence could either be classified as a complex, or a simple a correspondence.* The former case can be observed in Figure 8.1, where each implicit correspondence generated under a local perspective is formalised as a complex correspondence, e.g. $c_1 : \langle \mathsf{A_1}, \mathsf{C_2} \sqcap \neg \mathsf{B_2}, \equiv \rangle$. The latter, less frequent case occurs either (i) if the implicit correspondence of a defined entity in one ontology is aligned to another defined entity, such as $\mathsf{A_1}$ and $\mathsf{A_2}$ with the correspondence $c_1 : \langle \mathsf{A_1}, \mathsf{A_2}, \equiv \rangle$; (ii) or when a defined concept corresponds to a implicit synonym definition pattern (Section 3.4).

For example, given an ontology $\mathcal{O}$:

$$\mathcal{O} = \{\alpha_1 : \mathsf{A} \equiv \exists \mathsf{r}.\top, \alpha_2 : \mathsf{A} \sqsubseteq \mathsf{B}, \alpha_3 : \mathsf{B} \sqsubseteq \mathsf{A}\}$$

where concept A is explicitly defined by the axiom α_1, and implicitly defined by the axioms $\{\alpha_2, \alpha_3\}$, and there is an explicit correspondence $c_1 : \langle \mathsf{B}, \mathsf{X}, \equiv \rangle$ such that $\mathsf{X} \in \mathcal{O}'$, then A is implicitly mapped by the simple correspondence $c_2 : \langle \mathsf{A}, \mathsf{X}, \equiv \rangle$.

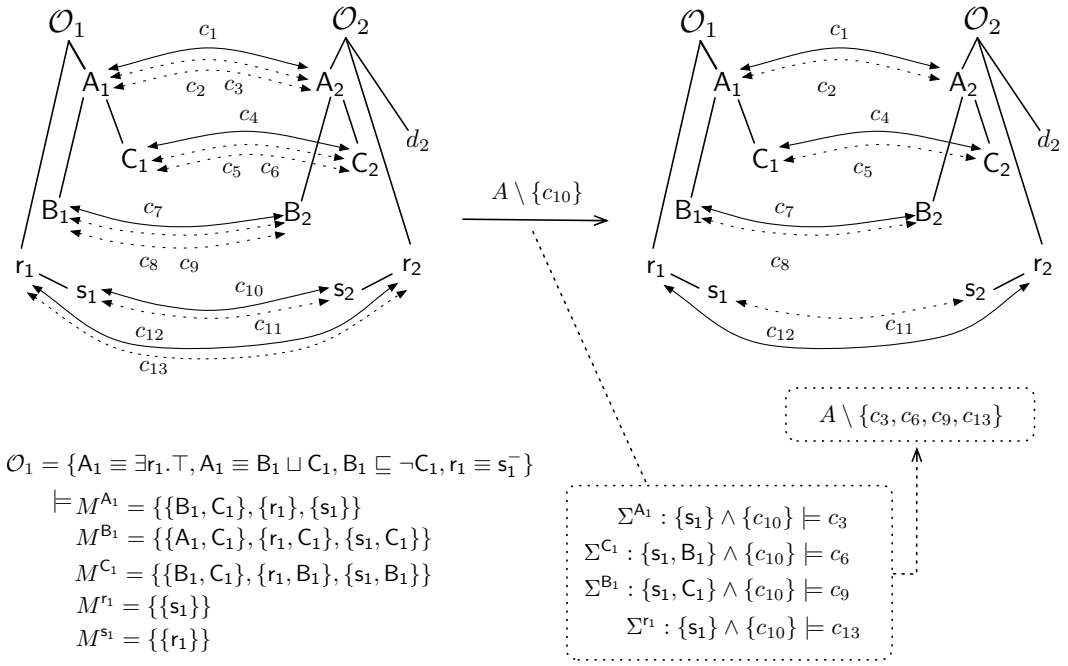


FIGURE 8.2: Entities of aligned ontologies mapped both explicitly and implicitly by multiple different correspondences, where M denotes the complete set of MDS of an entity. Simple correspondences are represented as normal, implicit correspondences as dashed arcs.

Definability-based Alignments. As shown in example depicted by Figure 8.1, some otherwise *unmapped* entities may become mapped by an alignment when definability is taken into account. Furthermore, some *already mapped* entities "gain" additional correspondences, as show by the next example. Let us consider Figure 8.2, where the entities of the aligned ontology pair $\mathcal{O}_1, \mathcal{O}_2$ are covered by multiple correspondences that correspond to the MDSs of defined entities. The concept A_1 is explicitly mapped by c_1, and implicitly mapped by two implicit correspondences c_2 and c_3. At this state, the alignment provides full coverage over the signature of $\mathcal{O}_1$. However, if the correspondence c_{12} which explicitly maps r_1 to r_2 is removed from the alignment, both roles remain mapped (although now only implicitly) because the inferred correspondence c_{13} depends only on the presence of c_{10} in the alignment ($\mathcal{O}_1, M^{\mathsf{r}_1} \models c_{13}$, where $c_{13} : \langle \mathsf{r}_1, \mathsf{s}_1^-, \equiv \rangle$), hence removing c_{12} does not reduce the coverage of the alignment. However, the removal effects some other implicit correspondences that are based on an MDS which contains the no longer explicitly mapped role r_1 (c_2, c_5, c_8). Thus the presence of definability-based correspondences potentially both potentially increases the number of covered entities of a given ontology, and reduces the loss in coverage, by considering definability-based correspondences that retain the expressive capacity of an alignment.

8.1.1 Issue with implicit correspondences

One issue with relying on implicit correspondences is that the aligned ontologies might disagree on the definitions (or even the definable status of a term). Let us consider the following scenario: there are two ontologies such that $\mathcal{O} = \{\mathsf{C} \equiv \mathsf{A} \sqcap \mathsf{B}\}$, $\mathcal{O}' = \{\mathsf{C}' \equiv \mathsf{A}' \sqcup \mathsf{B}'\}$, and an alignment A that maps $\mathcal{O}$ to $\mathcal{O}'$, where $A = \{c_1 : \langle \mathsf{A}, \mathsf{A}', \equiv \rangle, c_2 : \langle \mathsf{B}, \mathsf{B}', \equiv \rangle\}$. In this case C and C' are implicitly mapped to each other, however they disagree on the implicit correspondence due to the different definitions.

This problem might happen, however, it still makes sense to consider implicit correspondences as an "educated guess", because effectively automated ontology alignment is a best guess itself based on matching certain unique attributes of entities (e.g. names), hierarchical structures, background knowledge, etc..

8.2 Extending Coverage and Path-based Metrics

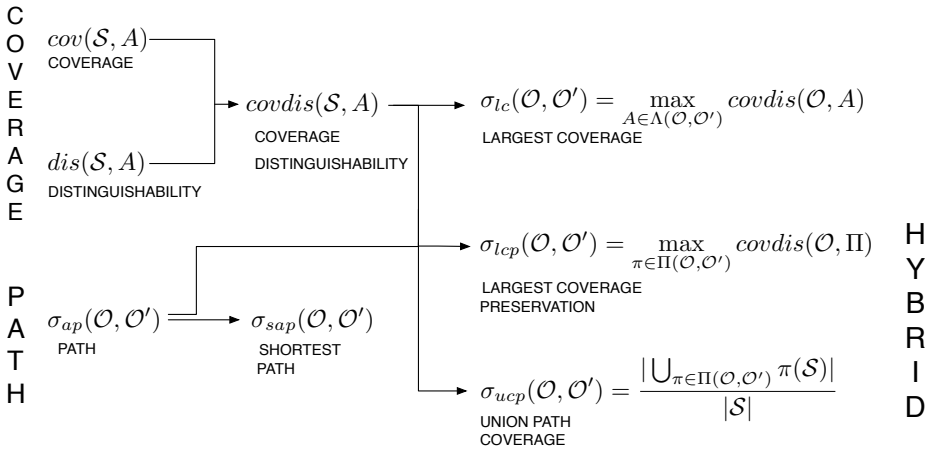


FIGURE 8.3: Topology of measures assessing ontology similarity in the alignment space. The two basic coverage-based metrics (coverage and distinguishability) and the path-based measure form a group of more complex (hybrid) metrics.

Classical ontology similarity measures [45] obtain similarity by directly comparing the content of ontologies, in contrast with alignment-based metrics [31] that quantify the proximity of ontologies with regards to how the ontologies are related by alignments. In addition to comparing ontologies, alignment-based measures can be used to compare *alignment quality* (without using a reference alignment as oppose to traditional evaluation metrics such as precision and recall[1].) by assessing the entities mapped by an alignment on two axes: *coverage* (or preservation) describes the number of entities mapped by an alignment, *distinguishability* (or separability) shows the number of matched entities that are kept distinct. These measures incorporate the notion of signature in order to focus their assessment on a particular subject matter (i.e. a signature

[1]Two alignments A, A' are compared with a reference alignment A_{GS}

of a TBox, query etc.). Furthermore, David *et al.* has also introduced a group of metrics that measure the ability and extent of information transfer from one ontology to another, by considering alignments as *paths* between (a network of) ontologies [31].

As previously demonstrated in Section 8.1, accounting for definability potentially improves simple alignments by entailing new correspondences. Albeit these implicit correspondences are entailed by an ontology, i.e. they are consequences of minimal definition signatures that are computed using internal ontological knowledge, once instantiated and published, they become part of the alignment space and in that sense independent of the source ontology. Therefore to accurately apply alignment-based metrics, in addition to explicit correspondences, implicit correspondences should be examined as well. However, traditional alignment-based ontology similarity metrics are not equipped to handle definability, thus in the following, we describe and extend these syntactic metrics in order to facilitate the measurement of definability-based correspondences.

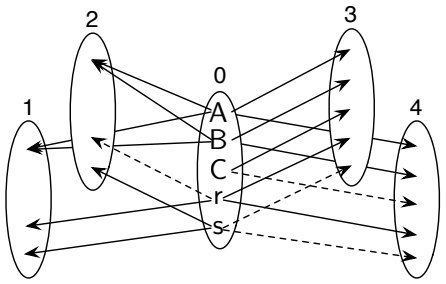
The three basic metrics (coverage, distinguishability and path) provide the base for more complex metrics. Figure 8.3 presents the metric topology: the combination of coverage-based measures form $covdis(\mathcal{S}, A)$, which accounts for both preservation and distinguishability, this is extended by the *largest covering alignment* (σ_{lcp}) metric, which identifies the best alignment (providing highest coverage and distinguishability w.r.t. a signature) within an alignment space (where $\mathcal{S}$ represents a signature and A denotes an alignment). *Path* (σ_{ap}) identifies the existence of a connection between two nodes in a multigraph formed by the alignment space (where ontologies are nodes and alignments are edges), whilst σ_{sap} shows the *shortest path*, which is the minimum number of nodes required to be visited in order to propagate information between two ontologies. Lastly, the hybrid measures provide more sophisticated assessment of an alignment space by evaluating alignment quality with respect to paths: *largest covering path* (σ_{lcp}) finds the path providing highest coverage and distinguishability, which facilitate cases when queries are executed on a single path; whilst *largest covering union path* (σ_{ucp}) identifies the combination (union) of best paths, in case a query is split into parts and run on more than one path, i.e. by using several ontologies and alignments.

In the following, we assume that all definability-based correspondences are named under a global perspective; and that correspondences can only describe permitted relations i.e. $\mathsf{r} \in \{\equiv, \sqsubseteq, \sqsupseteq, \perp\}$, thus the following definitions omit the notion of correspondence relation. Please note that all metrics are normalised, providing a real number within the range of $[0, 1]$.

8.2.1 Coverage-based Metrics

Coverage. *Alignment coverage* [31] (denoted as *cov*) is a syntactic metric that measures similarity with respect to a signature by counting the number of entities mapped *explicitly* by an alignment:

$$E_{exp} = \forall \mathsf{e}\{\mathsf{e} \in \mathcal{S} | \exists \left\langle \mathsf{e}, \mathsf{e}', r \right\rangle \in A\} \tag{8.1}$$



$\mathcal{O} = \{\mathsf{C} \equiv \mathsf{A} \sqcup \mathsf{B}, \mathsf{A} \sqsubseteq \neg\mathsf{B}, \mathsf{r} \equiv \mathsf{s}^{-}\}$

A	cov	d_cov	dis	d_dis	covdis	d_covdis
0-1	0.80	0.80	0.75	0.75	0.60	0.60
0-2	0.60	0.80	0.66	0.75	0.39	0.60
0-3	0.80	1.00	1.00	1.00	0.80	1.00
0-4	0.60	1.00	1.00	1.00	0.60	1.00

FIGURE 8.4: Comparing the default and definability-based coverage and distinguishability metrics in an alignment space, between ontology $\mathcal{O}_0$ and for other knowledge bases ($\mathcal{O}_1 - \mathcal{O}_4$). Explicit correspondences are represented as normal, implicitly correspondences are denoted as dashed arrows.

where $\mathcal{S}$ denotes the signature. Figure 8.4 presents an example, where the ontology $\mathcal{O}_0$ is mapped with four other ontologies; the coverage of the alignment $A_{0,3}$ is 4/5 with respect to $\mathcal{S} = \mathsf{Sig}(\mathcal{O}_0)$ because apart from the role s, all other entities are mapped explicitly. Definability-based coverage counts both explicitly and implicitly mapped entities, accounting for the fact that a given entity can simultaneously fall under both types (i.e. the numerator is the union of explicitly and implicitly mapped entities):

Definition 8.4 (Definability-based alignment coverage)**.** Given a set of ontology entities $\mathcal{S}$ of ontology $\mathcal{O}$, and an alignment $A \in \Lambda(\mathcal{O}, \mathcal{O}')$, the coverage of $\mathcal{S}$ by A is given by:

$$d\_cov(\mathcal{S}, A) = \frac{|E_{exp} \cup E_{imp}|}{|\mathcal{S}|}$$

where E_{exp} and E_{imp} denotes the set of explicitly and implicitly mapped entities of $\mathcal{S}$ by A, respectively.

Hence in Figure 8.4, the definability-based coverage of alignment $A_{0,3}$ is 5/5, because $\mathsf{s} \in \mathcal{O}_0$ is implicitly mapped, i.e. all entities of $\mathcal{O}_0$ are covered by the alignment.

Distinguishability. Alignments are not always injective, i.e. provide only a single correspondence for a given aligned entity, which could lead to ambiguity. Thus another important notion of alignment quality is the ability of an alignment to *preserve the difference between entities which are deemed different* in the source ontology. In other words, alignment distinguishability indicates the level of ambiguity that occurs in an alignment. For example, in Figure 8.4, the distinguishability of the alignment $A_{0,2}$ is 2/3 with respect to $\mathcal{S} = \mathsf{Sig}(\mathcal{O}_0)$, whereas $A_{0,4}$ (which provides the same level of coverage as $A_{0,2}$) gives total, 3/3 separability to entities of $\mathcal{S}$. The syntactic *alignment distinguishability* metric [31] (denoted as *dis*) counts the explicitly mapped target ontology entities:

$$\forall \mathsf{e}'\{\exists \langle \mathsf{e}, \mathsf{e}', r \rangle \in A \wedge \mathsf{e} \in \mathcal{S}\} \tag{8.2}$$

A definability-based correspondence maps an entity of an ontology to a named concept (or role) to another named entity, or to a complex concept (or role) of another ontology. In the latter case, it must be determined whether the *complex description* corresponds to a named entity under the target ontology, to avoid classifying it as a false positive, because some complex descriptions may denote the same thing (concept, or role).

Definition 8.5 (Definability-based alignment distinguishability)**.** Given a set of ontology entities $\mathcal{S}$ over an ontology $\mathcal{O}$, and an alignment $A \in \Lambda(\mathcal{O}, \mathcal{O}')$, the definability-based distinguishability (or separability) of $\mathcal{S}$ by A is given by:

$$d\_dis(\mathcal{S}, A) = \frac{|E'_{exp} \cup E'_{imp}|}{|E_{exp} \cup E_{imp}|}$$

where

- E_{exp} and E_{imp} denote the set of explicitly and implicitly mapped entities of $\mathcal{S}$, respectively;
- E'_{exp} denotes the set of explicitly mapped entities of $\mathcal{O}'$, and E'_{imp} denotes the set of implicitly mapped entities and complex concepts (or roles) of $\mathcal{O}'$;

by an alignment A.

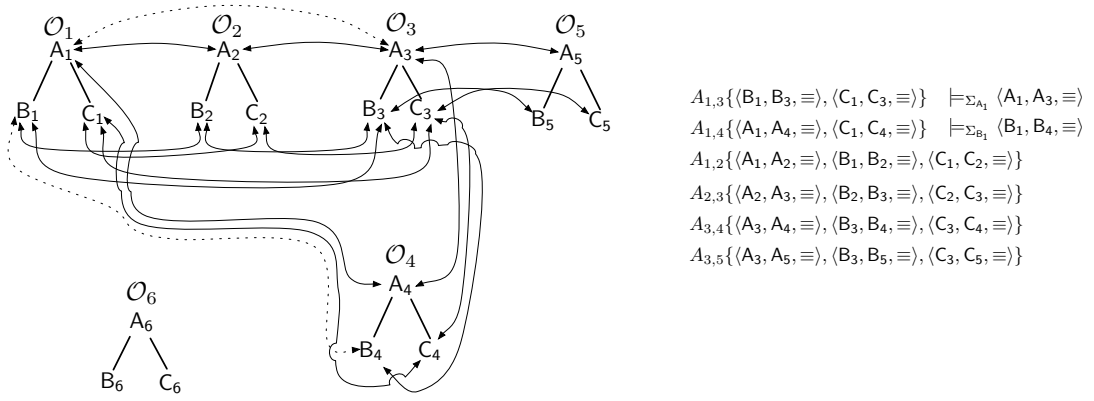
Therefore, by counting the entities and descriptions mapped by definability-based correspondences, the metric includes the accurate the number of separable entities of the target ontology, thus in Figure 8.4 the distinguishability of alignment $A_{0,2}$ is improved from 2/3 to 3/4.

Coverage-distinguishability. The *alignment coverage-distinguishability* [31] (denoted as *covdis*) metric simultaneously accounts for both coverage and distinguishability (it is calculated as the product of these metrics), thus it provides a more refined evaluation of alignment quality than its base metrics. For instance, in Figure 8.4, the best alignment is $A_{0,3}$ by achieving the highest *covdis* score in the target ontology. Furthermore, according to this joint metric, alignments $A_{0,1}$ and $A_{0,4}$ are equally useful to convey information from the source ontology $\mathcal{O}_0$. We extend this metric as follows:

Definition 8.6 (Definability-based alignment coverage-distinguishability)**.** Given a set of ontology entities $\mathcal{S}$ over an ontology $\mathcal{O}$, and an alignment $A \in \Lambda(\mathcal{O}, \mathcal{O}')$, the definability-based alignment coverage distinguishability of $\mathcal{S}$ by A is given by:

$$d\_covdis(\mathcal{S}, A) = d\_cov(\mathcal{S}, A) \cdot d\_dis(\mathcal{S}, A) = \frac{|E'_{exp} \cup E'_{imp}|}{|\mathcal{S}|}$$

(where $E_{exp}, E_{imp}, E'_{exp}, E'_{imp}$ are given by Definition 8.5).



	σ_{lc}	σ_{lcp}	d_σ_{lc}	d_σ_{lcp}
$\mathcal{O}_2$	1.00 ($A_{1,2}$)	1.00 ($A_{1,2}$)	1.00 ($A_{1,2}$)	1.00 ($A_{1,2}$)
$\mathcal{O}_3$	0.66 ($A_{1,3}$)	1.00 ($A_{1,2} \cdot A_{2,3}$)	1.00 ($A_{1,3}$)	1.00 ($A_{1,3}$)
$\mathcal{O}_4$	0.66 ($A_{1,4}$)	1.00 ($A_{1,2} \cdot A_{2,3} \cdot A_{3,4}$)	1.00 ($A_{1,4}$)	1.00 ($A_{1,4}$)
$\mathcal{O}_5$	0.00	1.00 ($A_{1,2} \cdot A_{2,3} \cdot A_{3,5}$)	0.00	1.00 ($A_{1,3} \cdot A_{3,5}$)

FIGURE 8.5: Comparing default (table top) and definability-based hybrid metrics (table bottom) in an alignment space, between $\mathcal{O}_1$ and five other ontologies ($\mathcal{O}_2 - \mathcal{O}_6$). Explicit correspondences are represented as normal, implicitly correspondences are denoted as dashed arrows.

Reevaluating the alignments with *d_covdis* changes the results, alignment $A_{0,3}$ is still the best, however, now $A_{0,4}$ is ranked equally as high, because both alignments provide full coverage and separability for the signature of $\mathcal{O}_0$ when definability is measured.

8.2.2 Path-based Metrics

A *path* [31] between two ontologies $\mathcal{O}$ and $\mathcal{O}$ in an alignment space, denoted as $\pi(\mathcal{O}, \mathcal{O}')$, is a finite sequence of alignments starting at $\mathcal{O}$ and finishing at $\mathcal{O}'$, where no intermediate ontology (or alignment) is included twice. The set of paths is denoted as $\Pi(\mathcal{O}, \mathcal{O}')$. Path-based measures ($\sigma_{ap}$: path existence, σ_{sap}: path length) are crude, as they only assess the fact whether there is a path between two ontologies (which could consists of only a single correspondence), and do not incorporate the notion of a signature. Accounting for definability does not change basic path-based measures because implicit correspondences can only be entailed by an already existing alignment thus definability provides no new paths, but potentially strengthens existing connections. For example, in Figure 8.5, which depicts an alignment space made up by five ontologies ($\mathcal{O}_1, \mathcal{O}_4$) and six different alignments, the path $\Pi(\mathcal{O}_1, \mathcal{O}_4)$ contains three different acyclic paths: $\pi_1 : A_{1,2} \cdot A_{2,3} \cdot A_{3,4}, \pi_2 : A_{1,3} \cdot A_{3,4}$ and $\pi_3 : A_{1,4}$. The shortest path is π_3, which is a direct alignment between $\mathcal{O}_1$ and $\mathcal{O}_4$. In this example, excluding $\mathcal{O}_6$, all ontologies are connected, hence it is possible to transfer information (to some extent) between any pair of ontologies $\mathcal{O}_i, \mathcal{O}_j \in (\Omega \setminus \{\mathcal{O}_6\})$.

8.2.3 Hybrid Metrics

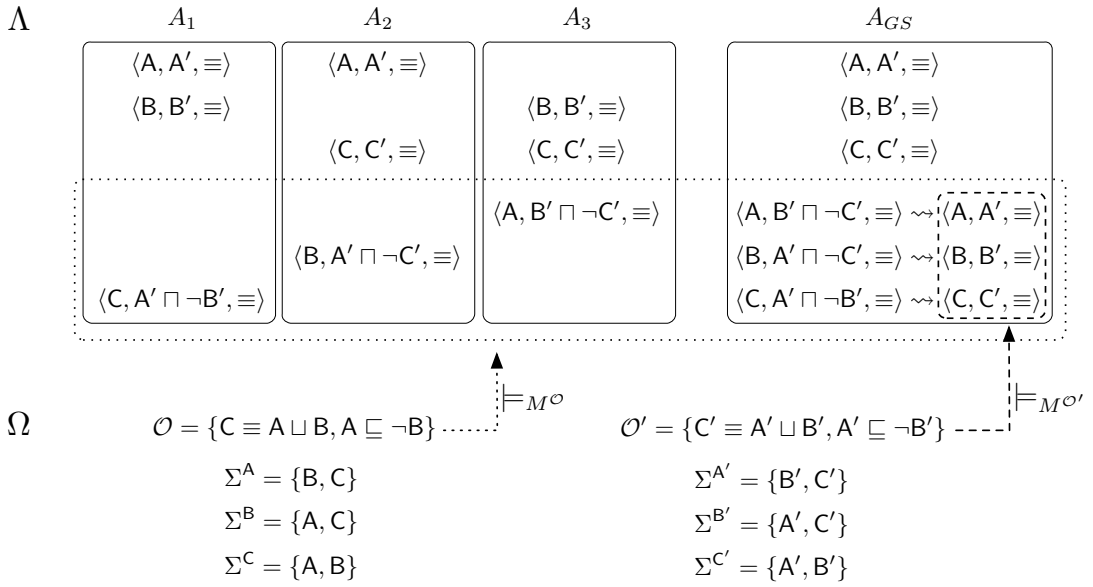
Hybrid measures incorporate the notion of paths into alignment evaluation, where a path itself is an alignment which is composed from basic alignments. As these measures are extensions of the perviously discussed basic metrics that were modified to account for definability, the two hybrid measures do not require any further modifications in order to be able to handle implicit correspondences.

In contrast with the *largest covering alignment* (σ_{lc}) metric which selects one alignment, the *largest covering path* (σ_{lcp}) widens the search space in identifying the highest quality alignment composition (or path) in the alignment space, for a given ontology pair. For example, in Figure 8.5, the only existing alignment between $\mathcal{O}_1$ and $\mathcal{O}_4$ ($A_{1,4}$) provides partial coverage (2/3) of $\mathcal{O}_1$ signature, whereas the path $\pi : A_{1,2} \cdot A_{2,3} \cdot A_{3,4}$ maps all of its entities. However, by considering implicit correspondences (d_σ_{lc}), a path is no longer necessary to achieve full coverage, as the alignment $A_{1,4}$ also maps all entities of $\mathcal{O}_1$. Furthermore, in the case of aligning $\mathcal{O}_1$ to $\mathcal{O}_5$, the shortest path without implicit correspondences is composed from three alignments ($\pi : A_{1,3} \cdot A_{2,3} \cdot A_{3,5}$), while by using definability-based correspondences this is reduced to a path consisting of only two alignments ($\pi : A_{1,3} \cdot A_{3,5}$).

The *largest covering union path* (σ_{ucp}) returns the composition of paths that provide the largest coverage and separability for a given signature, where the end point of the connection may involve more than one ontologies. Similarly to the largest covering path, the union path metric also benefits from accounting for definability, as it is potentially provides reduction of path lengths, and increase in alignment coverage.

8.3 Extending Precision and Recall

Classical precision and recall [47], and their harmonic mean, the F-measure, are de-facto ontology alignment evaluation metrics that assess the degree of conformance of produced alignments, generated by different matching systems, with regard to a reference alignment. These metrics are purely syntactic, as they only consider the asserted correspondences in an alignment. However, if the semantics of definability are taken into account, alignments potentially entail implicit correspondences, thus precision and recall metrics should also account for definability, in order to accurately measure alignment quality. Let us consider the example presented by Figure 8.6, where three produced alignments ($A_1 - A_3$), which map the entities of ontology $\mathcal{O}$ to the signature of $\mathcal{O}'$, are compared to a reference alignment. The recall score of alignment A_1 that represents the proportion of correct correspondences that are found is 0.66, because it contains 2 out of 3 correspondences that appear in the reference alignment A_{GS}, whilst its precision that represents the proportion of found correspondences that are correct is 1.00, as both correspondences in A_1 are correct. However, if the alignment contains definability-based correspondences: A_1 entails the implicit correspondence $\langle \mathsf{C}, \mathsf{A}' \sqcup \mathsf{B}', \equiv \rangle$, because C has a minimal definition signature Σ^{C} under $\mathcal{O}_1$, and A_1 contains explicit correspondences for all members



A	*Precision*	*Recall*	*d_Precision*	*d_Recall*
1	1.00	0.66	1.00	1.00
2	1.00	0.66	1.00	1.00
3	1.00	0.66	1.00	1.00

FIGURE 8.6: Evaluating syntactically different, but semantically equivalent alignments ($A_1 - A_3$) to a reference alignment (A_{GS}) using the default semantic precision and recall metric, and its definability-based variant.

of Σ^C (such that their relations can be reconciled to $\equiv$), thus $\mathcal{O}_1 \cup A_1 \models_{\Sigma^C} \langle C, A' \sqcup B', \equiv \rangle$ (where $\models_{\Sigma^C}$ denotes that a given alignment entails a definability-based correspondence supported by a particular MDS). This implicit correspondence is semantically equivalent to $\langle C, C', \equiv \rangle$, which appears in the reference alignment. Therefore, the argument can be made that if two alignments have equal precision and recall scores but different number of implicit correspondences, then the alignment which entails more correspondences and subsequently covers more entities is better. Therefore we extend the classic precision and recall to facilitate this notion. In contrast with classical precision and recall, the definability-based variant is calculated using the alignment closure which includes all implicit correspondences entailed by alignments:

Definition 8.7 (Definability-based alignment closure)**.** Given two aligned ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, and a corresponding alignment A, the definability-based closure of A, denoted as A^+, is the union of the normalised set of all *implicit correspondences* entailed by $(\mathcal{O}_1, A)$ and $(\mathcal{O}_2, A)$, and the *explicit correspondences* of A.

As a defined entity can have many different definitions, thus potentially many syntactically distinct correspondence can be present in a definability-based alignment, prior to evaluation alignments are required to be normalised, such that all but one semantically equivalent implicit correspondences of a given defined entity are removed from the

alignment. The following definition of definability-based precision and recall extends the classical formula by replacing both the evaluated alignment and the reference alignment with its definability-based closure:

Definition 8.8 (Definability-based precision and recall)**.** Given a reference alignment R, and a produced alignment A, the definability-based *precision* of A with respect to R is given by

$$d\_pr = \frac{|R^+ \cap A^+|}{|R^+|}$$

and the definability-based *recall* of A with respect to R is given by

$$d\_rc = \frac{|R^+ \cap A^+|}{|R^+|}$$

where R^+ and A^+ denote the reference alignment and a produced alignment closure, respectively.

Hence in the Figure 8.6 example, the extended metrics show that all three produced alignment provide 1.00 recall as they entail the same correspondences as the reference alignment.

8.3.1 Semantic Precision and Recall

Euzenat el al. have introduced *semantic precision and recall* to overcome several limitations of the classical approach [28, 44]. As the classical evaluation models are based on alignment syntax, they are not satisfactory because they ignore the semantics of ontologies, and the semantics of matching relations. As a result, semantically equivalent but syntactically different alignments are potentially assigned different precision and recall scores; this is illustrated by the next example:

Example 8.1. *Let us consider an ontology $\mathcal{O}_1$, a corresponding reference alignment A_{GS}, and two generated alignments A_1, A_2 that align $\mathcal{O}_1$ to another ontology such that*

- $\mathcal{O}_1 = \{\mathsf{C} \sqsubseteq \mathsf{A}\}$
- $A_{GS} = \{\langle \mathsf{A}, \mathsf{A}', \equiv \rangle\}$
- $A_1 = \{\langle \mathsf{A}, \mathsf{A}', \sqsubseteq \rangle, \langle \mathsf{A}', \mathsf{A}, \sqsubseteq \rangle\}, A_2 = \{\langle \mathsf{C}, \mathsf{A}', \sqsubseteq \rangle\}$

Although alignment A_1 is equivalent to the reference alignment, because $A \models \langle \mathsf{A}, \mathsf{A}', \equiv \rangle$, as a result of ignoring the alignment semantics, both precision and recall scores are zero. Furthremore under the rigid classical model, the otherwise correct correspondence $(\mathcal{O}_1, A_1) \models \langle \mathsf{C}, \mathsf{A}', \sqsubseteq \rangle$ of alignment A_2 is evaluated as a false-positive because the ontology semantics are ignored.

Semantic precision and recall address these flaws by comparing the *semantic closure* of alignments instead of the explicitly stated correspondences. Given two ontologies $\mathcal{O}$

and $\mathcal{O}'$, and an alignment A between these ontologies, the semantic closure of A (denoted as $Cn(A)$) consists of all correspondences that are entailed by the integrated ontology $\mathcal{O} \cup A \cup \mathcal{O}'$ [2]. An entailed correspondence is referred to as an $\alpha-$*consequence*. For instance, in Example 8.1, the correspondence $\langle \mathsf{A}, \mathsf{A}', \sqsubseteq \rangle$ is entailed (solely) by the alignment, whereas the correspondence $\langle \mathsf{C}, \mathsf{A}', \sqsubseteq \rangle$ is the consequence of the internal semantics of $\mathcal{O}_1$.

We make the argument that any definability-based correspondence is (by definition), an $\alpha-$*consequence*, therefore such correspondences should be considered in the semantic closure of alignments. Implicit correspondences are indeed consequences of the semantics of an ontology and its corresponding alignment, as they are based on the MDSs entities (i.e. semantic of ontologies), and are considered due to the availability of explicitly mapped entities, with compatible correspondence relations, (i.e. semantic of alignments) of particular definition signatures. Figure 8.6 illustrates this notion, for example the implicit correspondence $\langle \mathsf{C}, \mathsf{A}' \sqcup \mathsf{B}', \equiv \rangle$, which covers concept C, is an $\alpha-$consequence of alignment A_1 because C has a minimal definition signature Σ^{C} under $\mathcal{O}_1$, and A_1 contains explicit correspondences (with the same type of reconcilable relations) for all members of Σ^{C}, thus $\mathcal{O}_1 \cup A_1 \models_{\Sigma^{\mathsf{C}}} \langle \mathsf{C}, \mathsf{A}' \sqcup \mathsf{B}', \equiv \rangle$. We refer to a definability-based $\alpha-$consequence as a $\delta-$*consequence* and define it as follows:

Definition 8.9 ($\delta-$*consequence*)**.** Given an aligned ontology pair $\mathcal{O}$ and $\mathcal{O}'$, its corresponding alignment A, and the set of all minimal definition signatures $M^{\mathcal{O}}$ of defined entities of $\mathcal{O}$; a correspondence c is a $\delta-$*consequence* of $\mathcal{O}, \mathcal{O}'$ and A if and only if $\mathcal{O} \cup A \cup \mathcal{O}' \models_{M^{\mathcal{O}}} c$.

In order to incorporate *definability-based correspondences in semantic precision and recall calculations*, the closure of an alignment space should contain both all $\alpha-$, and all $\delta-$consequences. Since the introduction of the semantic precision and recall evaluation model, several variations [28, 50] have been proposed to address limitations of the original metrics[3], however, as the base notions of $\alpha-$consequence and closure remained the same, any variation can facilitate definability-based evaluation. One of the limitations is the *infinite closure space problem*, i.e. considering complex alignments results in a potentially infinite number or $\alpha-$consequences. This was resolved by restricting alignments to contain only simple correspondences [28]. However, definability-based correspondences are often formalised as complex correspondences, therefore to accommodate such correspondences, this constraint can be relaxed to considering complex alignments, but only permitting definability-based complex correspondences[4] and simple correspondences to be part of an alignment. Although the number of possible MDSs of a defined entity

[2] Please note that here we present a simplified version of these definitions, as this is sufficient in the context of this work. The complete description is grounded on the model theoretic semantics of alignments.

[3] These limitations were described and partially addressed in [28].

[4] Section 8.1 describes that a definability-based and a classical complex correspondence can be distinguished by checking whether the mapped anonym concept(s) (or role(s)) correspond to a named entity in either the target or source ontology.

is potentially exponential in the size of the ontology signature (the power set of the ontology signature), hence there could be a large number of implicit correspondences in a semantic alignment closure, the size of the closure space, which includes implicit correspondences, remains finite.

Computing $\delta-$consequence closures. A $\delta-$consequence can be computed in two ways, it is either entailed by an MDS and a particular set of simple correspondences, or it is inferred fron an MDS and a correspondence set containing definability-based complex correspondences. The former case is demonstrated in the example shown by Figure 8.6, while the latter is presented in the following example:

Example 8.2. *Let us consider an the ontology $\mathcal{O}_1$ and the alignment $A_{1,2}$ such that*

- $\mathcal{O}_1 = \{\mathsf{C} \equiv \mathsf{A} \sqcup \mathsf{B}, \mathsf{A} \sqsubseteq \neg\mathsf{B}\}$
- $A_{1,2} = \{c_1 : \langle \mathsf{A}, \mathsf{A}', \equiv \rangle, \mathsf{c_2} : \langle \mathsf{C}, \mathsf{A}' \sqcup \mathsf{B}', \equiv \rangle\}$

The correspondence $\langle \mathsf{B}, \mathsf{B}', \equiv \rangle$ is a $\delta-$consequence of $\mathcal{O}_1 \cup A_{1,2} \cup \mathcal{O}_2$ because

$$
\begin{aligned}
&(1)\ \mathcal{O}_1 \cup A_{1,2} \cup \mathcal{O}_2 \models \mathsf{C} \equiv \mathsf{A}' \sqcup \mathsf{B}' && \textit{(explicit correspondence)}\\
&(2)\ \mathcal{O}_1 \cup A_{1,2} \cup \mathcal{O}_2 \models \mathsf{A} \equiv \mathsf{A}' && \textit{(explicit correspondence)}\\
&(3)\ \mathcal{O}_1 \cup A_{1,2} \cup \mathcal{O}_2 \models \mathsf{C} \equiv \mathsf{A} \sqcup \mathsf{B}' && \textit{(from 1, 2)}\\
&(4)\ \mathcal{O}_1 \cup A_{1,2} \cup \mathcal{O}_2 \models \mathsf{B}' \equiv \mathsf{C} \sqcap \neg\mathsf{A} && \textit{(from 3)}\\
&(5)\ \mathcal{O}_1 \models \mathsf{B} \equiv \mathsf{C} \sqcap \neg\mathsf{A} && (\textit{i.e. } \Sigma^{\mathsf{B}} = \{\mathsf{A}, \mathsf{B}\})
\end{aligned}
$$

thus it holds that $\mathcal{O}_1 \cup A_{1,2} \cup \mathcal{O}_2 \models \mathsf{B} \equiv \mathsf{B}'$. Note that if the correspondence c_2 is replaced with its semantically equivalent variant $c_3 : \langle \mathsf{A} \sqcup \mathsf{B}, \mathsf{A}' \sqcup \mathsf{B}', \equiv \rangle$, the $\delta-$consequence $\langle \mathsf{B}, \mathsf{B}', \equiv \rangle$ is still entailed because

$$
\begin{aligned}
&(6)\ \mathcal{O}_1 \models \mathsf{C} \equiv \mathsf{A} \sqcup \mathsf{B}\\
&(7)\ \mathcal{O}_1 \cup A_{1,2} \cup \mathcal{O}_2 \models \mathsf{A} \sqcup \mathsf{B} \equiv \mathsf{A}' \sqcup \mathsf{B}' && \textit{(explicit correspondence)}\\
&(8)\ \mathcal{O}_1 \cup A_{1,2} \cup \mathcal{O}_2 \models \mathsf{C} \equiv \mathsf{A}' \sqcup \mathsf{B}' && \textit{(from 7)}
\end{aligned}
$$

the rest of the proof is the same as before, hence $\langle \mathsf{B}, \mathsf{B}', \equiv \rangle$ is a $\delta-$consequence, regardless of the syntactic form of the complex correspondence.

Both $\delta-$consequence computation cases require some form of reasoning. In the former case, first the complete set of MDSs needs to be computed, then a covered MDS (i.e. a definition signature whose entities are explicitly mapped) should be identified, followed by the generation of a corresponding definition axiom. In the later case, the right-hand side of an entity definition axiom, a description γ is required to be checked against named entities of the target ontology to find the entity whose definition it corresponds to, this is conducted as entailment check ($\mathcal{O} \models_? \mathsf{e} \equiv \gamma$, where $\mathsf{e} \in Sig(\mathcal{O})$).

As previously noted, a definability-based correspondence may have several different syntactic forms. For example, in Figure 8.6 the alignment A_2 entails the correspondence $\langle \mathsf{B}, \mathsf{C}' \sqcap \neg \mathsf{A}', \equiv \rangle$, which, under the corresponding aligned ontology $\mathcal{O} \cup A_2$, is equivalent to the correspondence $\langle \mathsf{C} \sqcap \neg \mathsf{A}, \mathsf{C}' \sqcap \neg \mathsf{A}', \equiv \rangle$, furthermore, under $\mathcal{O} \cup A_2 \cup \mathcal{O}'$ the former two correspondences are considered the same as $\langle \mathsf{B}, \mathsf{B}', \equiv \rangle$. Thus, depending on the privacy constraints of an ontology, an implicit correspondence may contain the defined entity (whose MDS entails the correspondence) such as the correspondence $\langle \mathsf{B}, \mathsf{C}' \sqcap \neg \mathsf{A}', \equiv \rangle$ where the entity name b is made public, or $\langle \mathsf{C} \sqcap \neg \mathsf{A}, \mathsf{C}' \sqcap \neg \mathsf{A}', \equiv \rangle$, where B is kept private. In order to resolve this issue, all $\delta-$consequences are *normalised* to a form where descriptions are replaced with named entities under a global view.

8.4 Empirical Evaluation

This chapter has introduced a new, definability-based ontology correspondence, which in contrast with traditional correspondences, is not a direct result of an ontology matching process but the entailment of considering existing correspondences and MDSs. Furthermore, the chapter extended several ontology alignment metrics, that measure different alignment properties in order to facilitate the evaluation of alignments that consider implicit correspondences. In this section, we empirically evaluate the following hypotheses about the implications of considering implicit correspondences in alignments in order to sample the potential benefits that such correspondences provide to semantic interoperability scenarios, using the original and the definability-based alignment evaluation metrics:

1. implicit correspondences potentially increase alignment coverage, as such correspondences may map otherwise uncovered (but defined) entities (Section 8.4.1);
2. alignments with implicit correspondences retain coverage at a better rate than traditional alignments, as entities can be mapped both directly and indirectly (Section 8.4.2);
3. implicit correspondences improve alignment compactness as the size can be reduced whilst maintaining coverage (Section 8.4.3).

Furthermore the last experiment (Section 8.4.4) set compares the classical, and the definability-based precision and recall metrics, in order to evaluate how implicit correspondences effect alignment quality.

Evaluation Corpus. The *evaluation corpus* was assembled from two OWL datasets used by the OEAI. Each corpus contains a set of heterogenous ontologies, and their corresponding alignments, generated by a number of different matching systems that competed in the OAEI evaluation challenge. Furthermore, each ontology includes a reference alignment which aims to include all possible correct correspondences that can be generated between a given aligned ontology pair. Table 8.1 presents a summary of the

Ontology	DL Expressivity	Axioms	$\lvert N_C \cup N_R \rvert$	$Def\%$	M_{avg}
Conference corpus					
cmt	$\mathcal{ALCIN}(\mathcal{D})$	226	86	51.16%	1.09
conference	$\mathcal{ALCHIF}(\mathcal{D})$	285	109	65.14%	1.54
confOf	$\mathcal{SIN}(\mathcal{D})$	196	57	15.79%	3.33
edas	$\mathcal{ALCOIN}(\mathcal{D})$	739	138	28.99%	2.80
ekaw	$\mathcal{SHIN}$	233	108	27.78%	1.00
iasted	$\mathcal{ALCIN}(\mathcal{D})$	358	182	17.58%	1.75
sigkdd	$\mathcal{ALEI}(\mathcal{D})$	116	70	28.57%	1.55
		307.57	**107.14**	**33.57%**	**1.87**
LargeBio corpus					
FMA_nci	$\mathcal{ALCN}(\mathcal{D})$	3828	3700	0.11%	1.00
FMA_snomed	$\mathcal{ALCN}(\mathcal{D})$	10293	10161	0.04%	1.00
NCI_fma	$\mathcal{ALC}$	9083	6552	29.98%	1.32
NCI_snomed	$\mathcal{ALCH}$	30411	24041	28.50%	1.39
SNOMED_fma	$\mathcal{ALER}$	20243	13431	21.41%	1.10
SNOMED_nci	$\mathcal{ALER}$	71042	51180	57.25%	1.09
		24150.00	**18177.50**	**22.88%**	**1.15**

TABLE 8.1: Evaluation corpus

corpus properties, showing the DL expressivity, the number of logical axioms, number of concepts and roles ($|N_C \cup N_R|$) in the ontology signature, the ratio of definable concepts and roles ($Def\%$), and the average number of different minimal definition signatures per defined entity (M_{avg}). The *Conference track* contains 7 small pairwise aligned ontologies which describe the conference organisation domain. The *Large biomedical ontology track* consists of 6 vast and semantically rich medical ontologies. Both datasets contain ontologies with varying level of definability, as shown by the ratio of defined entities and the number of MDSs per entity.

Experimental Framework The *experimental framework*, which facilitated the experiments presented in previous chapters was extended by the OntoSim [31] library[5] which implements the original coverage and path-based metrics. Furthermore the framework employed the Alignment API [30] to manage alignments, as well as to compute the classical, and the semantic precision and recall scores.

8.4.1 Experiment 1: Coverage Increase

The first set of experiments tested the hypothesis that implicit correspondences potentially increase alignment coverage, i.e. the number of concepts and roles which have a correspondence in an alignment, because such correspondences may map otherwise uncovered (but defined) concepts or roles. This was assessed by measuring and comparing the coverage of aligned ontologies, as obtained by the original and the definability-based metrics. The experiments were conducted over all available aligned ontology pairs of the Conference (21 pairs) and the LargeBio corpus (3 pairs), using the reference alignments, as well as the alignments produced by the different matchers. The main difference is that

[5] `http://ontosim.gforge.inria.fr/`

Ontology Pair	*Reference Alignment*			|Λ|	*All alignments, correct correspondences*			*All alignments, all correspondences*		
	|A|	*cov*	*+Abs*		|A|	*cov*	*+Abs*	|A|	*cov*	*+Abs*
Conference corpus										
cmt conference	15	14.05%	0.97%	10	6.00	5.85%	0.00%	16.30	14.77%	1.03%
cmt confof	16	19.90%	0.57%	10	5.80	7.21%	0.32%	12.90	14.24%	1.28%
cmt edas	13	11.63%	0.57%	10	8.50	7.61%	0.00%	17.90	14.98%	0.85%
cmt ekaw	11	11.44%	0.57%	10	5.50	5.72%	0.00%	14.70	13.86%	1.25%
cmt iasted	4	3.38%	0.00%	10	3.90	3.29%	0.00%	9.90	7.94%	0.40%
cmt sigkdd	12	14.61%	0.00%	10	8.90	10.84%	0.00%	14.20	16.13%	0.81%
conference confof	15	16.23%	1.08%	8	9.38	10.15%	0.14%	22.38	20.36%	1.47%
conference edas	17	12.47%	1.55%	10	9.30	6.82%	0.08%	23.40	15.78%	1.11%
conference ekaw	25	21.96%	1.22%	10	11.60	10.19%	0.33%	25.90	21.10%	0.85%
conference iasted	14	9.56%	1.09%	10	4.70	3.21%	0.28%	17.60	10.84%	1.28%
conference sigkdd	15	15.84%	1.06%	10	9.00	9.50%	0.58%	17.20	16.00%	0.76%
confof edas	19	19.05%	0.33%	10	10.30	10.33%	0.20%	22.00	21.17%	0.51%
confof ekaw	20	22.48%	0.00%	10	11.80	13.49%	0.00%	18.20	20.01%	0.68%
confof iasted	9	8.57%	0.28%	10	4.70	4.47%	0.28%	11.90	10.82%	0.34%
confof sigkdd	7	9.28%	0.00%	10	4.50	5.96%	0.00%	8.90	10.82%	0.13%
edas ekaw	23	18.37%	0.33%	10	10.80	8.62%	0.29%	26.30	18.47%	0.46%
edas iasted	19	11.46%	0.00%	10	7.90	4.76%	0.00%	20.20	10.75%	0.26%
edas sigkdd	15	14.64%	0.00%	10	7.50	7.32%	0.00%	15.50	12.70%	0.39%
ekaw iasted	10	7.48%	0.28%	10	6.50	4.86%	0.28%	18.40	12.90%	0.78%
ekaw sigkdd	11	12.33%	0.00%	10	7.40	8.30%	0.00%	16.20	15.71%	0.37%
iasted sigkdd	15	13.88%	0.00%	8	11.75	10.88%	0.00%	30.88	21.89%	0.80%
	14.52	**13.74%**	**0.47%**		**7.89**	**7.59%**	**0.13%**	**18.14**	**15.30%**	**0.75%**
LargeBio corpus										
FMA_n. NCI_f.	3024	57.74%	0.00%	20	1874.1	38.47%	0.00%	2572.75	50.31%	0.00%
FMA_s. SNOMED_f.	8941	71.59%	0.12%	18	4146.11	34.97%	0.02%	4912.17	40.54%	0.04%
SNOMED_n. NCI_ s.	18844	47.78%	0.78%	13	12470.38	35.65%	0.50%	13185.62	37.14%	0.58%
	10269.67	**59.04%**	**0.30%**		**6163.53**	**36.36%**	**0.17%**	**6890.18**	**42.67%**	**0.21%**

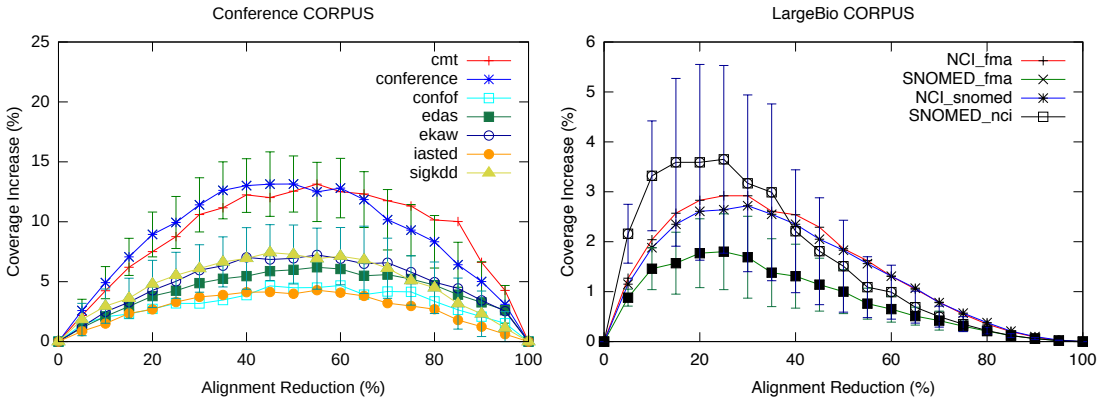
TABLE 8.2: Coverage

a reference alignment only contains correct correspondences, whereas produced alignments typically contain some incorrect correspondences as well. Thus a potential drawback of using definability-based correspondences is that incorrect base correspondences entail incorrect implicit correspondences. To assess this flaw, three different alignment settings were explored for each ontology pair. First, we have computed coverage values using only the reference alignment, in order to establish the *maximum correct coverage* that is possible to reach by a matcher. Next, we have used the produced alignments, but textitfiltered out the incorrect correspondences (i.e. false positives), to assess the *average correct coverage* achieved by the alignment systems. Filtered alignments were obtained by removing incorrect correspondences, i.e. those that neither appeared in the reference alignment (syntactically incorrect) nor were entailed by it (semantically incorrect). Finally, we have computed the coverage of the produced alignments, without filtering, to determine both the *average coverage* of ontologies and the *number of incorrectly entailed implicit correspondences*.

Results are presented in Table 8.2, which is divided into three parts, where each part shows the findings of one of the aforementioned alignment settings. The first part evaluates the maximum possible correct coverage, where $|A|$ denotes the size of a reference alignment, and *cov* indicates the average coverage ratio that an alignment provides for a given ontology pair (i.e. the coverage of all concepts and roles in the two ontologies). The coverage increase, which results from the potential presence of implicit correspondences in an alignment, is shown in terms of its absolute ($+Abs$) difference with the original coverage score. On average, a reference alignment provides 13.74% coverage for the ontologies in the Conference corpus, and 59.04% in the LargeBio corpus. In the Conference corpus, out of 21 alignments, 13 entail some implicit correspondences, resulting in an average 0.47% (absolute) increase in coverage per ontology. In the LargeBio corpus, out of 3 alignments, 2 had some implicit correspondences, which increased the coverage by 0.30%, on average. In terms of entities, definability-based correspondences covered an additional 1.43 entities in the small (Conference corpus), and 92.52 in the large ontologies.

The middle part of Table 8.2 presents the average correct coverage, where the column $|\Lambda|$ denotes the number of different alignments for a given aligned ontology pair, and $|A|$ denotes the average size of the alignments. In comparison with the maximum scores, the average size of produced alignments is about half of the reference alignment, thus the average alignment coverage has decreased significantly in both datasets, in terms of absolute difference by 6.15% (and 55.24%, in relative difference) in the Conference, and by 22.68% (i.e. 61.55%) in the LargeBio corpus. As a result, the number of implicit correspondences and the implicitly covered entities has also decreased, to less than a third of the maximum, 0.13% in the Conference, and by almost a half, to 0.17% in the LargeBio corpus.

The right part of Table 8.2 shows the coverage values obtained by using *unfiltered*



(a) (b)

FIGURE 8.7: Coverage retention in the Conference (*a*) and LargeBio (*b*) corpus.

alignments. In the Conference corpus, both the original and the definability-based coverage exceed the maximum possible correct value, thus more than half of the implicit correspondences, entailed by invalid base correspondences, were incorrect. In contrast with the LargeBio corpus, the number of incorrect implicit correspondences was considerably less, due to the fact that the size of the unfiltered alignments were not much larger than the filtered ones, i.e. contained more correct correspondences.

Although the distinguishability metric was also evaluated, we omit the results, because in almost every case (i.e. an instance of an implicit correspondence) distinguishability increased at the same rate as coverage, i.e. the implicit correspondence described a relation between a source ontology entity, and a description of the target ontology, not a named entity.

8.4.2 Experiment 2: Coverage Retention

Implicit correspondences provide coverage for certain defined entities of an aligned ontology signature. Such entities can be already mapped by the asserted alignment (i.e. explicitly covered), or can be otherwise unmapped and become (implicitly) covered as a result of emerging implicit correspondences. As shown in the previous section, the latter case increases the alignment coverage. In this section, we evaluate the hypothesis, that alignments with definability-based correspondences retain expressive capacity at a better rate than traditional alignments. Without considering implicit correspondences if a correspondences is removed from a given alignment, the alignment coverage typically decreases (unless there are more than one correspondences covering a particular entity, which potentially leads to ambiguity), whereas with implicit correspondences if an entity is simultaneously covered both explicitly and implicitly, removing its explicit correspondence does not effect the coverage because the entity remains still covered by an implicit correspondence.

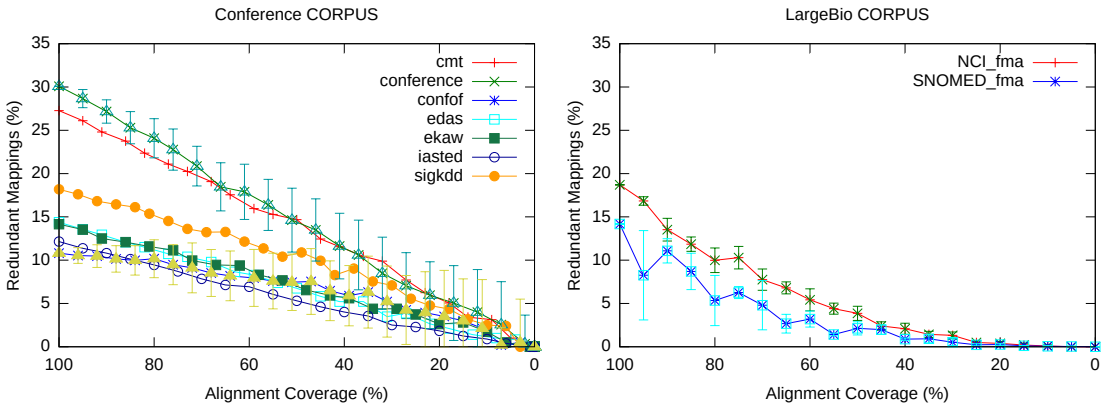
In order to verify this claim, we have conducted the following experiment: for each ontology, we have *generated an artificial alignment* which provided full coverage over

the ontology signature, then removed randomly selected correspondences and compared the original, and the definability-based coverage. The available alignments (OAEI) were insufficient to comprehensibly assess coverage retention, because these experiments required a large number of different alignments in every size between a full alignment and an empty set, thus we have used generated alignments. Albeit the alignments are artificial, this evaluation scenario can still be considered as a realistic setting, because when an ontology is aligned with its version, or with a closely related knowledge base, the resulting alignment would cover the majority of both ontologies signature. The experiment was carried out over 21 test cases (different alignment sizes) for each ontology, whereby the number of removed correspondences was increased by 5% (w.r.t. the original alignment size). Due to the random correspondence removal, each test case was repeated 100 times[6], thus the experiment measured the overall average coverage, and the standard deviation.

Figure 8.7 (a) and (b) present the results of the experiments conducted over the Conference and the LargeBio corpus, respectively. A baseline was generated by using the original coverage metric, this is an almost monotonically decreasing function, as removing one correspondence typically reduces the coverage of a single entity. The x-axis shows the alignment reduction, i.e. the ratio of removed correspondences, starting with a full alignment (0%) to an empty set (100%). The y-axis shows not the total alignment coverage, but the (absolute) difference between the definability-based coverage (which is the result of the presence of implicit correspondences) and the baseline (which is equivalent to the x-axis, thus it is omitted from the graph). As expected, in both datasets, implicit correspondences provide additional coverage, thus aligned ontologies considering implicit correspondences indeed retain expressive capacity better than traditional alignments. Ontologies with more defined entities can achieve better coverage retention than lesser defined ones, for instance, the *conference* ontology, which has the largest percentage of defined entities (65.14%, Table 8.1) exhibits the highest retention rate in the Conference corpus, while the *iasted* ontology has the lowest retention rate in the corpus because this also has the lowest ratio of defined entities (17.58%). The same trend can be observed in the LargeBio corpus[7]. While the Conference corpus shows an even distribution of coverage increase, the LargeBio corpus is right-skewed. This can be explained by the fact that ontologies in the former corpus, on average, have almost twice as many MDSs per defined entity, than in the LargeBio corpus, thus as the alignment reduction increases it is much less likely that LargeBio ontology entities retain coverage by implicit correspondences. For the purpose of readability the standard deviation is only shown for the ontologies that received the best, and the worst increase in coverage. The rate of standard deviation follows the rate of the coverage increase in both datasets,

[6]Several different repetition counts were tested in order to establish, by comparing their relative standard deviation, that 100 repetition was sufficient for both datasets.

[7]Please note, that in this corpus, the two ontologies with very low definability yielded no coverage increase, hence these were omitted from the graph.



(a) (b)

FIGURE 8.8: Alignment compactness in the Conference (*a*) and LargeBio (*b*) corpus.

i.e. it is the largest at peak coverage increase, and smallest when coverage increase is at its minimum.

8.4.3 Experiment 3: Compactness

The third experiment tested the hypothesis that implicit correspondences potentially improve the compactness of alignments, i.e. some part of a given alignment can be discarded without losing any coverage, thus the cardinality of an alignment can be reduced, making it more compact and therefore more valuable in semantic interoperability. Alignment compactness is equivalent to the notion of identifying a *minimal cover set for the explicit alignment coverage*, as a carefully selected smaller subset of an alignment, by employing both explicit and implicit coverage, may provide the same coverage as the whole alignment. The data of this experiment was obtained during the empirical evaluation of minimal cover sets (Section 7.4.2), where the experiment assessed the cover set reduction achieved by approximated minimal cover sets, thus here, the inverse of cover set reduction, compactness illustrates the benefits of utilising minimal cover sets in the ontology alignment context. Similarly to coverage retention, compactness experiments used generated alignments that initially covered the full signature, at each iteration, the alignment size was reduced by 5% increments (with randomly selected correspondences), where each iteration was repeated 100 times. For each test case, we first computed the explicit coverage provided by the alignment, then computed the minimal set of entities that provided the same coverage as the simple alignment, and measured the number correspondences that could be discarded due to the compactness increase provided by implicit correspondences.

Figure 8.8 (c) and (d) presents alignment compactness in the Conference and the LargeBio corpus, respectively. The x-axis shows the original coverage value provided by an alignment, with respect to the ontology signature starting with a full alignment (providing 100% coverage) down to an empty set (0% coverage). The y-axis shows the ratio of redundant correspondences in the alignment (those that can be discarded

whilst maintaining coverage). The baseline is (nearly) equivalent to the x-axis, because without applying definability, no correspondences can be removed from the alignment without loss of coverage, hence it is omitted from the graph. The results confirm that using implicit correspondences indeed results in more compact alignments, where the rate of achievable compactness depends on the definability of the particular ontology, and the composure of a given alignment, i.e. in general, alignments of better defined ontologies can be made more compact than the alignments of less defined ontologies. Regardless of the ontology size, the rate of compactness can be significant. For example, for all small ontologies in the Conference corpus, all alignments that covered 60% of the signature, were reducible by least 5%. Both datasets show the same trend: as the size of the alignment decreases, the rate of achievable compactness declines as well. Moreover, the standard deviation grows monotonically as the alignment shrinks, because smaller alignments are least likely to contain the set of explicit correspondences which entails implicit ones.

8.4.4 Experiment 4: Precision and Recall

The purpose of this experiment was to compare the classical, and the definability-based precision and recall metrics, in order to evaluate how considering implicit correspondences effect alignment quality. The experiment tested the hypothesis that considering implicit correspondences increase recall but potentially decrease precision as some implicit correspondences could be entailed by incorrect correspondences. In addition, the experiment aimed to correlate the coverage and the alignment quality achieved by the different matching systems to explore any relation between coverage, and precision and recall.

The experiment computed the classical and the definability-based average precision (pr, d_pr), recall (rc, d_rc) and f-measure ($FM1$, d_FM1) scores, for each matching systems that produced alignments for the 21 ontology pairs of the Conference corpus. The left and middle partitions of Table 8.3 present the results, where $|A|$ denotes the average size of alignments produced by a given approach, and $|R|$ denotes the average cardinality of the reference alignments. In comparison with classical precision and recall (shown in the left partition), the definability-based metrics (middle partition) shows worse performance both in terms precision and recall, for all matcher systems. However, this decrease is consistent over all matchers, as their performance ranking remains the same (ordered by the f-measure values). The alignment recall also decreases, because the produced alignments entail fewer implicit correspondences than the reference alignment.

The right partition of Table 8.3 shows several matcher rankings, ordered by different metrics, where *REF* denotes the reference alignment. The original (cov) and the definability-based coverage (d_cov) is showing nearly the same matcher ranking (only the bottom two matcher system order is different), furthermore, this confirms that both coverage metrics correspond to alignment cardinality, i.e. in general, more correspondences mean larger coverage. However, larger coverage does not necessarily mean better

	Original Precision & Recall					*Def.-based Precision & Recall*					*Matcher Ranking*				
Matcher	$\|A\|$	$\|R\|$	*pr*	*rc*	$FM1$	$\|A\|$	$\|R\|$	*d_pr*	*d_rc*	*d_FM*1	*cov*	*d_cov*	*cov*(cr)	*d_cov*(cr)	$FM1$
AML	10.95		85.91%	65.12%	72.79%	11.38		-3.11%	-4.39%	-4.05%	AOT	AOT	REF	REF	REF
LogML	9.95		75.51%	52.61%	60.68%	11.29		-6.83%	-3.35%	-4.95%	Maas	Maas	Maas	Maas	AML
LogM	10.21		82.82%	59.80%	68.18%	10.63		-2.80%	-4.15%	-3.79%	REF	REF	AML	AOT	LogM
Maas	34.14		29.13%	68.27%	40.24%	41.67		-4.77%	-4.68%	-5.42%	AOTL	AOTL	AOT	AML	LogMC
AOT	62.95	14.52	15.59%	65.74%	24.66%	73.90	15.86	-2.33%	-4.29%	-3.31%	AML	AML	LogM	LogM	XMap
AOTL	14.67		46.61%	48.73%	44.99%	15.71		-2.57%	-3.00%	-2.91%	LogM	LogML	LogMC	LogMC	RSD
LogMC	9.74		82.21%	57.48%	66.50%	10.05		-2.47%	-3.96%	-3.63%	LogML	LogM	LogML	LogML	LogML
RSD	8.33		83.67%	50.04%	61.23%	9.24		-6.84%	-3.07%	-4.50%	LogMC	LogMC	XMap	XMap	OMR
OMR	9.10		73.67%	48.99%	57.58%	10.00		-5.38%	-3.05%	-4.10%	RSD	RSD	RSD	RSD	AOTL
XMap	8.14		88.44%	52.09%	64.27%	8.48		-3.43%	-3.26%	-3.71%	XMap	OMR	OMR	OMR	Maas
	17.82		**66.36%**	**56.89%**	**56.11%**	**20.24**		**-4.05%**	**-3.72%**	**-4.04%**	OMR	XMap	AOTL	AOTL	AOT

TABLE 8.3: Matcher evaluation in the Conference corpus

alignment quality, as it is shown by comparing the coverage and the f-measure ($FM1$) rankings. For instance, according to coverage, the top three matchers are in fact the worst three systems in terms of alignment quality. In spite of this fact, comparing the original ($cov(cr)$) and the definability-based coverage ($d\_cov(cr)$) that was computed by using only correct correspondences, it can observed that the two matchers with the lowest $FM1$ score provide the highest correct coverage after the reference alignment (which provides the achievable maximum correct coverage).

8.5 Summary and Conclusions

This chapter explored a new, definability-based ontology correspondence and shown that considering MDSs under alignments is an a posteriori alignment aggregation process, moreover, extended several ontology alignment evaluation metrics to facilitate the evaluation of such correspondences. Furthermore, by comparing alignments with the original and the extended metrics, the evaluation confirmed the hypothesis that implicit correspondences can potentially increase the coverage, coverage retention and the compactness of alignments. However, the evaluation also suggests, that coverage does not necessarily correspond to precision as incorrect base correspondences lead to incorrect implicit correspondences. Nonetheless, experiments have also shown, that, by using only correct simple correspondences, coverage can be increased without effecting precision; therefore produced alignments should be filtered priori to considering implicit correspondences, this procedure is commonly performed by knowledge-based agents that impose a threshold over the confidence values (i.e. the optional meta-data of a correspondence which is assigned internally by the producing matcher system) to filter out potentially low quality correspondences.

Part IV

Synopsis

Chapter 9

Conclusions and Future Work

This thesis introduced the notion of *minimal definition signatures* (MDSs) and explored their computation in Description Logics ontologies. The contribution in this thesis proposes practical algorithms that permit exploiting of definability in scenarios where ontologies need to be reconciled (i.e. aligned or matched) in order to overcome semantic interoperability. MDSs effectively make use of implicit definitions hence a defined entity can be removed without semantic loss. The notion of MDS was shown to be sufficient to support ontology matching, alignment evaluation and ontology alignment negotiation in those cases where a complete or even a partial set of MDSs can be used to enrich existing alignments. This chapter summarises the main findings of this thesis, discusses the overall conclusions, and addresses limitations and outstanding issues of the presented work (Section 9.1); moreover, it outlines some directions for future work in the computation and application areas of MDSs (Section 9.2).

9.1 Summary and Conclusions

The overall goal of the research presented is to support and improve ontology alignment negotiation, a recent but established research area which concerns the generation of mutually acceptable alignments between collaborating systems (agents), to facilitate communication in dynamic environments. Independent agents typically adhere to different ontologies, and therefore to heterogeneous knowledge models. This heterogeneity hinders or even precludes knowledge-based interactions. In order to resolve heterogeneity, ontologies need to be aligned, however, no single ontology matching approach is necessarily suitable for all matching scenarios, furthermore, the resulting alignments are typically incomplete, providing only a partial coverage of an ontology vocabulary [124].

Definability and MDSs. The basic idea behind the research conducted in this PhD thesis is to exploit implicit definability by finding and using the definition signatures. A definition signature is an entity set from which an entity is implicitly definable under the ontology which fixes the meaning of the definable entity thus it can be removed without semantic loss.

Ten Cate and colleagues presented a method to determine whether a particular signature is sufficient to support implicitly definability of a given concept or role [136]. As implicit definability check works in languages that do not accept Beth definability, therefore Beth definability was not crucial for the practical applications of this work (please note that if a given DL accepts Beth definability then it means that whenever implicit definitions are found, explicit definitions can be generated). The contribution presented in this thesis focusses on obtaining definition signatures. Definition signatures may contain redundant members and in the worst case their size could be almost as large as the size of the ontology signature, thus we introduce the notion of signature minimality and Minimal Definition Signatures (MDSs).

Given that only defined entities have MDSs, the first step in obtaining MDSs for an entity is to determine whether such entity is defined, either implicitly or explicitly. Whilst checking explicit definability is an inexpensive process that checks the existence of an axiom in a TBox, verifying implicit definability is potentially a computationally expensive process, since it depends on the complexity of the entailment check for a given DL flavour. A second contribution of this thesis is an optimised implicit definability check. This optimisation is obtained by: (*i*) testing explicit definability prior to implicit, as the former is a syntactic notion; (*ii*) reducing the search space by applying modularisation; (*iii*) modularisation for undefined entities often results in an empty module, thus for such entities the expensive implicit definability check can be avoided.

The implicit definability check permits us to find different MDSs for the same concept or role description. However, the MDSs should be ultimately validated either by a human or by some verification process. For this reason it is useful to know not only the MDS, but also a set of axioms whose entailment justifies implicit definability. The third contribution of this thesis extends the implicit definability check and computes justifications, i.e. minimal sets of axioms that are sufficient for entailments to hold. These justifications can be used as evidential support for MDSs.

Computing MDSs. The process of determining single MDS is potentially a computationally expensive process, because in the worst case the number of candidate signatures that need to be explored is the size of power set of the ontology signature (excluding the the defined entity itself). The algorithm for computing single MDSs presented in this thesis was designed to address the high complexity, and performs a three stage definability computation process, which employs numerous MDS computation algorithms and optimisation heuristics, thus providing an efficient way to compute in practice all MDSs of the defined entities. In addition, to further reduce the complexity of definability computation steps, the process employs modularisation, which as the empirical evaluation suggests, typically provides a considerably smaller search space for entities to include in the signature when compared to the original ontology and can be efficiently computed.

Computing a single MDS is achieved by iteratively reducing an input signature, where each step performs a call to the oracle to validate whether the input signature still implicitly defines the entity in question, under the given ontology. However this is a potentially costly process if we consider that each oracle call itself may take exponential time in the number of axioms in the ontology (the oracle call complexity varies, depending on the expressivity of the DL language); thus, in addition to an approach, which makes linear number (w.r.t. the size of the input signature) of calls to the oracle (in the number of axioms in the ontology where each oracle call may itself take exponential time in the number of axioms in the ontology), a single-entity pruning approach was also developed, which at the best case makes logarithmic number (w.r.t. the size of the input signature) of calls to the oracle (in the number of axioms in the ontology where each oracle call may itself take exponential time in the number of axioms in the ontology), by applying a multi-entity pruning, divide and conquer strategy to reduce the input definition signatures into MDSs. While multi-entity pruning typically performs significantly better than the former approach, when the number of redundant signature entity members is higher than the required entities the upper bound of the number of calls to the oracle is the double of the former. Thus, by identifying the edge cases, the overall MDS computation process harmonises the use of the two approaches.

Computing all MDSs of defined entities is achieved by first identifying a set of pairwise disjoint MDSs, then iteratively expanding the obtained MDSs until the set is complete. Disjoint MDSs are obtained by iteratively extracting one MDS at the time from an input signature; this step is optimised for explicitly defined entities, by first examining the signature of their explicit definition axioms (when such axioms are present). While establishing the type of definability and computing the disjoint MDSs of a defined entity makes polynomial number of calls to the oracle, obtaining the complete set of MDSs requires exponential number of calls to the oracle as it requires iterating through, and testing each subset (i.e. the candidate signatures) of a power set of the ontology signature (excluding the defined entity itself).

The motivating scenario for our investigation was the negotiation of ontology alignments, therefore this thesis also aimed to establish whether, in practice, the use of MDSs would contribute to ontology alignment in particular, and ontology engineering in general. For this purpose the thesis conducted an empirical investigation that assesses the prevalence and the extent of definability over a wide range of OWL ontologies. Furthermore, the evaluation studied the behaviour of the proposed definability computation process, in terms of run time taken for each of the stages necessary to compute the MDSs, compared various MDS computation approaches, and analysed the impact of modularisation to definability computation. The evaluation has shown that (i) out of the DL ontologies considered in the large and diverse evaluation corpus, MDSs may occur regardless of the employed DL language, the size of an ontology, the conceptualised domain of interest, or its origin (source of creation); although it is more likely to occur in more expressive, and semantically richer ontologies;

(*ii*) establishing the definability status of an entity is a feasible for most real-world ontologies; (*iii*) computing a set of disjoint MDS is also feasible, however, the larger, more expressive ontologies take much longer to compute than the smaller, less expressive ones; (*iv*) obtaining the complete set of MDSs is feasible for most smaller ontologies, as the exponential complexity is grounded on the union of existing MDSs, which is typically small (we introduced a 20 element limit on the maximum union size, i.e. at most 2^{20} candidate sets were examined during expansion), however, the process potentially takes considerable amount of time for larger, or more expressive ontologies; (*v*) the definability computation process is highly parallelizable (i.e. it can be performed separately for each entity in the ontology signature), which improves feasibility of computing for larger ontologies.

Applications of MDSs. The empirical investigation included obtaining MDSs for numerous ontologies, and validating them by computing the corresponding justifications. As implicit definitions are often not straightforward to recognise and interpret, a number of definition patterns were identified, by studying the composition of MDSs (their cardinality, and the type and number of their member entities) together with their justifications (their size, and the type of their constituent axioms). Please note that this was not a comprehensive study. Although for the bigger picture of the thesis, the exact definitions are not important, it is still interesting to know what kind of explicit definitions one can find for implicitly definable concept and role names. The patterns aim to generalise the frequent forms of creating definitions, however, these are non-exhaustive. In addition to the validation and the interpretation of definability, the identifiable definition patterns permit a heuristic-based definition axiom generation, where the generation of an explicit definition of a defined entity is achieved by processing a given MDS and a justification according to an inference rule. As the evaluation suggests, given an ontology, often the majority of definability cases can be categorised by the identified patterns, thus a definition axiom can be generated.

The prerequisite for any knowledge-based task is that it must be covered by the signature which is available for the party that is performing a task. Considering definability supports ontology signatures to express more than the constituent, asserted entities, because an entity which has MDS(s) can be removed without semantic loss as the meaning of the definable entity is wholly fixed by the terms of its definition. Hence by considering implicit coverage (which is entailed by the complete set of MDSs of a given ontology, or one of its module), a potentially smaller task signature can be identified than by only considering explicit coverage of the task signature. Finding the minimal cover set poses a challenge, as the complete set of MDSs needs to be known, and all combinations of such definition signatures are required to be explored, for each entity in question. Thus, this thesis introduced the minimal cover set problem, and presented an approximation approach that provides non-redundant cover sets, that, whilst not minimal, are still considerably smaller than a cover set obtained by only explicit coverage.

As a final contribution, this thesis has introduced definability-based correspondences that permits exploiting the notion of MDSs in semantic interoperability, where knowledge-based interactions between heterogenous ontologies are typically supported by alignments. Such correspondences emerge when a defined entity is wholly fixed by the terms of its definition and the terms in the definition are mapped by the available alignment; thus implicit definability entails a new type of correspondence, based on the MDSs of entities in the aligned ontologies and the available alignment. Furthermore, as the existing evaluation approaches were insufficient to accurately assess definability-based alignments, several alignment evaluation metrics were extended to facilitate measuring such correspondences, both with and without reference alignments. The evaluation confirmed the hypothesis that considering definability-based correspondences can potentially increase the coverage, coverage retention and the compactness of alignments; moreover, definability-based alignment metrics provide more accurate measure than the classical metrics.

9.2 Future Work

While the presented work has made considerable advancement in obtaining (some or the complete set of) MDSs, and explored several application areas which benefit from using MDSs, there are several limitations which can constrain the practical usability of the computation of MDS, especially in dynamic environments. In addition, the research has opened up new questions and MDS application areas. In the following, we discuss some of the remaining challenges and sketch out possible further contributions emerging from this thesis:

- *Role definability check and MDSs.* The role definability check (Section 3.3.5) presented an issue with false positives when the TBox was used during the computation and therefore it was restricted to using the RBox and to definition signature consisting only role names. This issue could be further investigated to ensure that the definability status check of roles does not exclude certain cases (e.g. use of nominals to define roles) and that all role MDSs can be computed.

- *Definition patterns and heuristic-based rewriting.* The patterns presented in Chapter 3 are not meant to be exhaustive, as they are not proven to represent all cases of possible definitions, thus a more fine-grade classification could be developed. Moreover, the research conducted in this thesis showed that the identified patterns facilitate a rewriting approach that allows the reformulation of concepts and roles based on heuristics; this could also be explored further to aid definition axiom generation in scenarios where MDSs are available, and the used DL languages accept Beth definability which permits generating explicit definitions based on implicit ones. This axiom generation could be used when there are no other means (i.e. a specific rewriting service) available.

- *Minimal cover computation.* The presented approach is not guaranteed to find the minimal (smallest) cover set, but only find suboptimal solutions through a greedy approach that builds the cover set incrementally, and performs a random selection whenever there is more than one viable option available. An alternative approach could build a tree model of the execution, thus allowing for some backtracking that supports the exploration of new cover sets, without degrading the polynomial complexity.

- *Ontology alignment with definability.* In addition to classical techniques such as computing string similarity scores of entity labels, recent ontology matching systems identify complex relations between entities (or entity sets) of different ontologies by using semantic techniques, such as reasoning (e.g. LogMap [81]), or patterns [121, 133]. However, when aligning ontologies that are conceptualised with different level of granularity entities can be rewritten through complex definitions might not be matched appropriately (i.e. with imprecise correspondences). Several approaches make use of explicit definitions to identify complex relations, but the ability of identifying implicit definitions could widen the search space and hence increase the chances of aligning complex concepts (and roles).

  Definability based alignments can also support opportunistic (anytime) alignments, that are generated dynamically through some form of negotiation rather than using traditional a-priori approaches, as described in Section 6.5.

  This thesis claims that systems engaged in a negotiation process are better equipped to cooperatively establish a mutually acceptable alignment, when these alignments are determined using the rewriting approaches proposed in this thesis. The arguments in favour of our claims are manifold:

  (*i*) Definability-based evaluation metrics provide systems with more accurate measures to determine whether an alignment provides the necessary coverage to achieve a particular task (i.e. align the whole ontology or just those entities necessary to formulate a message or query, etc.);

  (*ii*) By instantiating definability-based correspondences, the alignment coverage may be increased, which potentially widens the set of possible knowledge-based tasks that a system agent can support;

  (*iii*) By computing minimal signature coverage, systems may generate minimal mutual alignments. In contrast with the approach presented in Doran et al. [39], that uses modularisation, based on a task signature, in order to reduce the space of candidate correspondences, minimal covers can potentially provide an even smaller search space, as the module signature may contain several cover sets.

  (*iv*) The availability of alternative cover sets might support systems in adhering to privacy and confidentiality policies, whereby only some entities or axioms in their ontologies can be disclosed and shared, whereas others are considered confidential (or commercially sensitive) and should not be readily shared.

- *Agent coalition formation with definability.* Ontology alignment negotiation is a potentially computationally costly process (Section 6.5). Assuming a multi-agent system populated by knowledge-based agents, an agent may have multiple options to find a suitable partner for collaboration. Prior to the negotiation process, by making use of definability-based alignment evaluation metrics, agents can establish an initial ranking of their peers according to their similarity (based on the available alignments).

Bibliography

[1] Patrick Arnold, *Semantic Enrichment of Ontology Mappings: Detecting Relation Types and Complex Correspondences*, Grundlagen von Datenbanken **1020** (2013), 34–39.

[2] Patrick Arnold and Erhard Rahm, *Enriching ontology mappings with semantic relations*, Data & Knowledge Engineering **93** (2014), 1–18.

[3] Manuel Atencia, Alexander Borgida, Jérôme Euzenat, Chiara Ghidini, and Luciano Serafini, *A formal semantics for weighted ontology mappings*, The Semantic Web–ISWC 2012 (2012), 17–33.

[4] Manuel Atencia and Marco Schorlemmer, *An interaction-based approach to semantic alignment*, Web Semantics: Science, Services and Agents on the World Wide Web **12** (2012), 131–147.

[5] Franz Baader, *The Description Logic Handbook: theory, implementation, and applications*, Cambridge University Press, 2003.

[6] Franz Baader, Sebastian Brandt, and Carsten Lutz, *Pushing the EL envelope*, The International Joint Conference on Artificial Intelligence (IJCAI), vol. 5, 2005, pp. 364–369.

[7] Franz Baader and Werner Nutt, *The Description Logic Handbook*, ch. Basic description logics, pp. 43–95, Cambridge University Press, 2003.

[8] Sidney C Bailin and Walt Truszkowski, *Ontology negotiation between intelligent information agents*, The Knowledge Engineering Review **17** (2002), no. 01, 7–19.

[9] Trevor JM Bench-Capon, *Persuasion in practical argument using value-based argumentation frameworks*, Journal of Logic and Computation **13** (2003), no. 3, 429–448.

[10] Amaia Bernaras, Iñaki Laresgoiti, and Jose Corera, *Building and Reusing Ontologies for Electrical Network Applications'*, The European Conference on Artificial Intelligence (ECAI), PITMAN, 1996, pp. 298–302.

[11] Tim Berners-Lee, James Hendler, Ora Lassila, et al., *The semantic web*, Scientific American **284** (2001), no. 5, 28–37.

[12] Evert W Beth, *On Padoa's method in the theory of definition*, Indagationes Mathematicae **15** (1953), 330 – 339.

[13] Alexander Borgida and Luciano Serafini, *Distributed description logics: Assimilating information from peer sources*, J. Data Semantics **1** (2003), 153–184.

[14] Willem Nico Borst, *Construction of engineering ontologies for knowledge sharing and reuse*, Universiteit Twente, 1997.

[15] L Bos and K Donnelly, *SNOMED-CT: The advanced terminology and coding system for eHealth*, Stud Health Technol Inform **121** (2006), 279–290.

[16] Paolo Bouquet, Antonia Dona, Luciano Serafini, and Stefano Zanobini, *ConTeXtualized local ontology specification via CTXML*, Working Notes of the AAAI-02 workshop on Meaning Negotiation. Edmonton (Canada), 2002.

[17] Paolo Bouquet, Fausto Giunchiglia, Frank Van Harmelen, Luciano Serafini, and Heiner Stuckenschmidt, *C-owl: Contextualizing ontologies*, International Semantic Web Conference, Springer, 2003, pp. 164–179.

[18] Paolo Bouquet, Bernardo Magnini, Luciano Serafini, and Stefano Zanobini, *A SAT-based algorithm for context matching*, International and Interdisciplinary Conference on Modeling and Using Context, Springer, 2003, pp. 66–79.

[19] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati, *Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family*, Journal of Automated Reasoning **39** (2007), no. 3, 385–429.

[20] Diego Calvanese, Giuseppe De Giacomo, Daniele Nardi, and Maurizio Lenzerini, *Handbook of automated reasoning*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, 2001, pp. 1581–1634.

[21] Thomas Cerqueus, Sylvie Cazalens, and Philippe Lamarre, *Reducing the semantic heterogeneity of unstructured p2p systems: a contribution based on a dissemination protocol*, Transactions on Large-Scale Data-and Knowledge-Centered Systems VII, Springer, 2012, pp. 62–95.

[22] Vinay K Chaudhri, Adam Farquhar, Richard Fikes, Peter D Karp, and James P Rice, *OKBC: A programmatic foundation for knowledge base interoperability*, Innovative Applications of Artificial Intelligence Conferences (AAAI/IAAI), 1998, pp. 600–607.

[23] Michelle Cheatham, Zlatan Dragisic, Jérôme Euzenat, Daniel Faria, Alfio Ferrara, Giorgos Flouris, Irini Fundulaki, Roger Granada, Valentina Ivanova, Ernesto Jiménez-Ruiz, Patrick Lambrix, Stefano Montanelli, Catia Pesquita, Tzanina Saveta, Pavel Shvaiko, Alessandro Solimando, Cássia Trojahn dos Santos, and

Ondrej Zamazal, *Results of the Ontology Alignment Evaluation Initiative 2015*, Proceedings of the 10th International Workshop on Ontology Matching, 2015, pp. 60–115.

[24] Paula Chocron and Marco Schorlemmer, *Attuning ontology alignments to semantically heterogeneous multi-agent interactions*, ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016), 2016, pp. 871–879.

[25] Thomas H Cormen, *Introduction to algorithms*, MIT press, 2009.

[26] Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler, *Modular reuse of ontologies: Theory and practice*, Journal of Artificial Intelligence Research (2008), 273–318.

[27] Jérôme David and Jérôme Euzenat, *Comparison between ontology distances (preliminary results)*, Springer, 2008.

[28] Jérôme David and Jérôme Euzenat, *On fixing semantic alignment evaluation measures*, Proc. 3rd ISWC workshop on ontology matching (OM), No commercial editor., 2008, pp. 25–36.

[29] Jérôme David, Jérôme Euzenat, and Jason J Jung, *Experimenting with ontology distances in semantic social networks: methodological remarks*, Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on, IEEE, 2012, pp. 2915–2920.

[30] Jérôme David, Jérôme Euzenat, François Scharffe, and Cássia Trojahn Dos Santos, *The Alignment API 4.0*, Semantic web journal **2** (2011), no. 1, 3–10.

[31] Jérôme David, Jérôme Euzenat, and Ondřej Šváb-Zamazal, *Ontology similarity in the alignment space*, The Semantic Web–ISWC 2010, Springer, 2010, pp. 129–144.

[32] Daniel Defoe, *Robinson Crusoe*, W. Taylor, United Kingdom, April 1719.

[33] Chiara Del Vescovo, *The modular structure of an ontology: Atomic Decomposition and its applications*, Ph.D. thesis, University of Manchester, 2013.

[34] Chiara Del Vescovo, Pavel Klinov, Bijan Parsia, Uli Sattler, Thomas Schneider, and Dmitry Tsarkov, *Syntactic vs. Semantic Locality: How Good Is a Cheap Approximation?*, The Workshops on Modular Ontologies, 2012.

[35] Robin Dhamankar, Yoonkyong Lee, AnHai Doan, Alon Halevy, and Pedro Domingos, *iMAP: discovering complex semantic matches between database schemas*, Proceedings of the 2004 ACM SIGMOD international conference on Management of data, ACM, 2004, pp. 383–394.

[36] Kevin Donnelly, *SNOMED-CT: The advanced terminology and coding system for eHealth*, Studies in health technology and informatics **121** (2006), 279.

[37] Paul Doran, *Ontology modularization: Principles and practice*, Ph.D. thesis, University of Liverpool, 2009.

[38] Paul Doran, Terry R Payne, Valentina Tamma, and Ignazio Palmisano, *Deciding agent orientation on ontology mappings*, International Semantic Web Conference, Springer, 2010, pp. 161–176.

[39] Paul Doran, Valentina Tamma, Ignazio Palmisano, and Terry R Payne, *Applying Ontology Modularization to Argumentation over Ontology Correspondences in MAS*, The Eighth International Conference on Autonomous Agents and Multiagent Systems (AAMAS'09), 2009.

[40] Paul Doran, Valentina Tamma, Terry R Payne, and Ignazio Palmisano, *Dynamic selection of ontological alignments: a space reduction mechanism*, Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09), 2009, pp. 2028–2033.

[41] Paul Doran, Valentina Tamma, Terry R Payne, and Ignazio Palmisano, *Using ontology modularization for efficient negotiation over ontology correspondences in MAS*, International Workshop on Argumentation in Multi-Agent Systems, Springer, 2009, pp. 236–255.

[42] Paul Doran, Valentina Tamma, Terry R Payne, and Ignazio Palmisano, *Flexible agreement mechanism for dynamic meaning negotiation*, Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1, International Foundation for Autonomous Agents and Multiagent Systems, 2010, pp. 1599–1600.

[43] Jérôme Euzenat, *Towards a principled approach to semantic interoperability*, Proc. IJCAI 2001 workshop on ontology and information sharing (Seattle, United States), Proc. IJCAI 2001 workshop on ontology and information sharing, No commercial editor., August 2001, euzenat2001b, pp. 19–25.

[44] Jérôme Euzenat, *Semantic Precision and Recall for Ontology Alignment Evaluation*, Proceedings of the 20th International Joint Conference on Artifical Intelligence (San Francisco, CA, USA), IJCAI'07, Morgan Kaufmann Publishers Inc., 2007, pp. 348–353.

[45] Jérôme Euzenat, Carlo Allocca, Jérôme David, Mathieu d'Aquin, Chan Le Duc, and Ondrej Svab-Zamazal, *Ontology distances for contextualisation*, Tech. report, IST NeOn, 2009.

[46] Jérôme Euzenat, Christian Meilicke, Heiner Stuckenschmidt, Pavel Shvaiko, and Cássia Trojahn, *Ontology Alignment Evaluation Initiative: Six Years of Experience*, Journal on Data Semantics XV (Stefano Spaccapietra, ed.), Springer-Verlag, Berlin, Heidelberg, 2011, pp. 158–192.

[47] Jérôme Euzenat and Pavel Shvaiko, *Ontology Matching, Second Edition*, Springer, 2013.

[48] Jérôme Euzenat, Heiner Stuckenschmidt, and Mikalai Yatskevich, *Introduction to the ontology alignment evaluation 2005*, Proceedings of K-Cap, Citeseer, 2005, pp. 61–71.

[49] Dieter Fensel, Ian Horrocks, Frank Van Harmelen, Deborah McGuinness, and Peter F Patel-Schneider, *OIL: Ontology infrastructure to enable the Semantic Web*, IEEE intelligent systems **16** (2001), no. 2, 38–45.

[50] Daniel Fleischhacker and Heiner Stuckenschmidt, *A practical implementation of semantic precision and recall*, Complex, Intelligent and Software Intensive Systems (CISIS), 2010 International Conference on, IEEE, 2010, pp. 986–991.

[51] David Geleta, Terry R Payne, and Valentina Tamma, *An Investigation of Definability in Ontology Alignment*, The 20th International Conference on Knowledge Engineering and Knowledge Management (EKAW2016), Springer, 2016, To Appear.

[52] David Geleta, Terry R. Payne, and Valentina Tamma, *Computing Minimal Definition Signatures in Description Logic Ontologies*, Tech. report, Department of Computer Science, University of Liverpool, 2016.

[53] David Geleta, Terry R. Payne, and Valentina Tamma, *Computing Minimal Signature Coverage for Description Logic Ontologies*, Tech. Report ULCS-16-XXX, Department of Computer Science, University of Liverpool, 2016.

[54] David Geleta, Terry R Payne, and Valentina Tamma, *Definability-based Ontological Correspondences and Alignment Evaluation Metrics*, IEEE Transactions on Knowledge and Data Engineering (TKDE), 2016, In Submission.

[55] David Geleta, Terry R Payne, and Valentina Tamma, *Minimal Coverage for Ontology Signatures*, 13th OWL: Experiences and Directions Workshop and 5th OWL reasoner evaluation workshop (OWLED – ORE 2016), Springer, 2016, In Submission.

[56] Michael R Genesereth, Richard E Fikes, et al., *Knowledge interchange format-version 3.0: reference manual*, (1992).

[57] Pierdaniele Giaretta and N Guarino, *Ontologies and knowledge bases towards a terminological clarification*, Towards very large knowledge bases: knowledge building & knowledge sharing **25** (1995), 32.

[58] Pascal Gillet, Cassia Trojahn, Ollivier Haemmerlé, and Camille Pradel, *Complex correspondences for query patterns rewriting*, Proceedings of the 8th International Conference on Ontology Matching, vol. 1111, CEUR-WS. org, Kent State UNiversity, 2013, pp. 49 – 60.

[59] Fausto Giunchiglia, Aliaksandr Autayeu, and Juan Pane, *S-Match: an open source framework for matching lightweight ontologies*, Semantic Web **3** (2012), no. 3, 307–317.

[60] Fausto Giunchiglia, Vincenzo Maltese, and Aliaksandr Autayeu, *Computing minimal mappings between lightweight ontologies*, International Journal on Digital Libraries **12** (2012), no. 4, 179–193.

[61] Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich, *S-Match: an algorithm and an implementation of semantic matching*, European semantic web symposium, Springer, 2004, pp. 61–75.

[62] Birte Glimm, *Querying description logic knowledge bases*, Ph.D. thesis, University of Manchester, 2007.

[63] Birte Glimm, Ian Horrocks, Boris Motik, Giorgos Stoilos, and Zhe Wang, *HermiT: an OWL 2 reasoner*, Journal of Automated Reasoning **53** (2014), no. 3, 245–269.

[64] A. Gómez-Pérez, *Knowledge sharing and reuse*, The Handbook of Applied Expert Systems (J. Liebowitz, ed.), CRC Press LLC, Boca Raton, FL, 1998, pp. 10.1–10.36.

[65] Bernardo Cuenca Grau, Zlatan Dragisic, Kai Eckert, Jérôme Euzenat, Alfio Ferrara, Roger Granada, Valentina Ivanova, Ernesto Jiménez-Ruiz, Andreas Oskar Kempf, Patrick Lambrix, Andriy Nikolov, Heiko Paulheim, Dominique Ritze, François Scharffe, Pavel Shvaiko, Cássia Trojahn, and Ondřej Zamazal, *Results of the ontology alignment evaluation initiative 2013*, Proceedings of the 8th International Conference on Ontology Matching - Volume 1111 (Aachen, Germany, Germany), OM'13, CEUR-WS.org, 2013, pp. 61–100.

[66] Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel-Schneider, and Ulrike Sattler, *OWL 2: The Next Step for OWL*, Web Semant. **6** (2008), no. 4, 309–322.

[67] Benjamin N Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker, *Description logic programs: combining logic programs with description logic*, Proceedings of the 12th international conference on World Wide Web, ACM, 2003, pp. 48–57.

[68] Thomas R Gruber, *A translation approach to portable ontology specifications*, Knowledge acquisition **5** (1993), no. 2, 199–220.

[69] Thomas R Gruber, *Toward principles for the design of ontologies used for knowledge sharing*, International journal of human-computer studies **43** (1995), no. 5, 907–928.

[70] Nicola Guarino, *Formal ontology and information systems*, Proceedings of FOIS, vol. 98, 1998, pp. 81–97.

[71] Tom Heath and Christian Bizer, *Linked data: Evolving the web into a global data space*, vol. 1, Morgan & Claypool Publishers, 2011.

[72] Eva Hoogland et al., *Definability and interpolation: Model-theoretic investigations*, Institute for Logic, Language and Computation, 2001.

[73] Matthew Horridge, *Justification Based Explanation in Ontologies: A Thesis Submitted to the University of Manchester for the Degree of Doctor of Philosophy in the Faculty of Engineering and Physical Sciences*, Ph.D. thesis, University of Manchester, 2011.

[74] Matthew Horridge and Sean Bechhofer, *The OWL API: A Java API for OWL ontologies*, Semantic Web **2** (2011), no. 1, 11–21.

[75] Matthew Horridge, Jonathan M. Mortensen, Bijan Parsia, Ulrike Sattler, and Mark A. Musen, *A Study on the Atomic Decomposition of Ontologies*, Proceedings of the 13th International Semantic Web Conference - Part II (New York, NY, USA), ISWC '14, Springer-Verlag New York, Inc., 2014, pp. 65–80.

[76] Matthew Horridge, Bijan Parsia, and Ulrike Sattler, *The owl explanation workbench: A toolkit for working with justifications for entailments in owl ontologies*, 2015.

[77] Matthew Horridge and Peter F. Patel-Schneider, *OWL 2 Web Ontology Language Manchester Syntax (Second Edition)*, `https://www.w3.org/TR/owl2-manchester-syntax/` (visited on 22 September 2016).

[78] Ian Horrocks et al., *DAML+OIL: A Description Logic for the Semantic Web*, IEEE Data Eng. Bull. **25** (2002), no. 1, 4–9.

[79] Wei Hu, Jianfeng Chen, Hang Zhang, and Yuzhong Qu, *Learning complex mappings between ontologies*, Joint International Semantic Technology Conference, Springer, 2011, pp. 350–357.

[80] Shangpu Jiang, Daniel Lowd, and Dejing Dou, *Ontology Matching with Knowledge Rules*, International Conference on Database and Expert Systems Applications, Springer, 2015, pp. 94–108.

[81] Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau, *Logmap: Logic-based and scalable ontology matching*, International Semantic Web Conference, Springer, 2011, pp. 273–288.

[82] Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, Yujiao Zhou, and Ian Horrocks, *Large-scale Interactive Ontology Matching: Algorithms and Implementation*, ECAI, vol. 242, 2012, pp. 444–449.

[83] Ernesto Jiménez-Ruiz, Terry R Payne, Alessandro Solimando, and Valentina Tamma, *Avoiding Alignment-based Conservativity Violations through Dialogue*, The 12th OWL: Experiences and Directions Workshop (OWLED), 2015.

[84] Ernesto Jiménez-Ruiz, Terry R Payne, Alessandro Solimando, and Valentina Tamma, *Limiting Consistency and Conservativity Violations through Negotiation*, 15th International Conference on Principles of Knowledge Representation and Reasoning (KR 16), 2016.

[85] H. Kitakami, Y. Mori, and M. Arikawa, *An intelligent system for integrating autonomous nomenclature databases in semantic heterogeneity*, Proceedings of Database and expert system applications (DEXA'96) (R.R. Wagner and H. Thoma, eds.), Springer, 1996, pp. 187–196.

[86] M. Klein, *Combining and relating ontologies: an analysis of problems and solutions*, Proceedings of the IJCAI'01 workshop on Ontologies and Information Sharing (A. Gómez-Pérez, M. Gruninger, H. Stuckenschmidt, and M. Uschold, eds.), 2001, pp. 53–62.

[87] Boris Konev, Dirk Walther, and Frank Wolter, *The logical difference problem for description logic terminologies*, International Joint Conference on Automated Reasoning, Springer Berlin Heidelberg, 2008, pp. 259–274.

[88] Markus Krötzsch, Frantisek Simancik, and Ian Horrocks, *A Description Logic Primer*, arXiv preprint arXiv:1201.4089, 2012.

[89] Loredana Laera, Ian Blacoe, Valentina Tamma, Terry Payne, Jerôme Euzenat, and Trevor Bench-Capon, *Argumentation over ontology correspondences in MAS*, Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems, ACM, 2007, p. 228.

[90] Loredana Laera, Valentina Tamma, Jérôme Euzenat, Trevor Bench-Capon, and Terry Payne, *Arguing over ontology alignments*, Proceedings of the 1st International Conference on Ontology Matching-Volume 225, CEUR-WS. org, 2006, pp. 49–60.

[91] Loredana Laera, Valentina Tamma, Jérôme Euzenat, Trevor Bench-Capon, and Terry Payne, *Reaching agreement over ontology alignments*, International Semantic Web Conference, Springer, 2006, pp. 371–384.

[92] Loredana Laera, Valentina Tamma, Jérôme Euzenat, Trevor Bench-Capon, and Terry R Payne, *Agents arguing over ontology alignments*, Fourth European Workshop on Multi-Agent Systems, 2006.

[93] Alexander Maedche, Boris Motik, Nuno Silva, and Raphael Volz, *Mafra—a mapping framework for distributed ontologies*, International Conference on Knowledge Engineering and Knowledge Management, Springer, 2002, pp. 235–250.

[94] Sabine Massmann, Salvatore Raunich, David Aumüller, Patrick Arnold, and Erhard Rahm, *Evolution of the COMA match system*, Proceedings of the 6th International Conference on Ontology Matching-Volume 814, CEUR-WS. org, 2011, pp. 49–60.

[95] Nicolas Matentzoglu, Samantha Bail, and Bijan Parsia, *A corpus of OWL DL ontologies*, Proc. DL'13, 2013.

[96] Nicolas Matentzoglu, Samantha Bail, and Bijan Parsia, *A Snapshot of the OWL Web*, The Semantic Web – ISWC 2013 (Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann, JosianeXavier Parreira, Lora Aroyo, Natasha Noy, Chris Welty, and Krzysztof Janowicz, eds.), Lecture Notes in Computer Science, vol. 8218, Springer Berlin Heidelberg, 2013, pp. 331–346 (English).

[97] Brian McBride, *Jena: A semantic web toolkit*, IEEE Internet computing **6** (2002), no. 6, 55.

[98] Deborah L McGuinness, *Ontologies come of age*, Spinning the semantic web: bringing the World Wide Web to its full potential (2005), 171.

[99] Deborah L McGuinness, Frank Van Harmelen, et al., *OWL web ontology language overview*, W3C recommendation **10** (2004), no. 10, 2004.

[100] Christian Meilicke, Heiner Stuckenschmidt, and Andrei Tamilin, *Repairing ontology mappings*, 2007.

[101] George A Miller, *WordNet: a lexical database for English*, Communications of the ACM **38** (1995), no. 11, 39–41.

[102] L. Morgenstern, *Inheritance comes of age: Applying nonmonotonic techniques to problems in industry*, Artificial Intelligence **103** (1998), 1–34.

[103] Boris Motik and Ian Horrocks, *OWL datatypes: Design and implementation*, Springer, 2008.

[104] Mark A Musen, *National Cancer Institute Thesaurus*, Encyclopedia of Systems Biology (2013), 1492–1492.

[105] Natalya F Noy, Nigam H Shah, Patricia L Whetzel, Benjamin Dai, Michael Dorf, Nicholas Griffith, Clement Jonquet, Daniel L Rubin, Margaret-Anne Storey, Christopher G Chute, et al., *BioPortal: ontologies and integrated data resources at the click of a mouse*, Nucleic acids research (2009), gkp440.

[106] Lorena Otero-Cerdeira, Francisco J Rodríguez-Martínez, and Alma Gómez-Rodríguez, *Ontology matching: A literature review*, Expert Systems with Applications **42** (2015), no. 2, 949–971.

[107] Christos M. Papadimitriou, *Computational complexity*, Addison-Wesley, Reading, Massachusetts, 1994.

[108] Terry R Payne and Valentina Tamma, *A dialectical approach to selectively reusing ontological correspondences*, International Conference on Knowledge Engineering and Knowledge Management, Springer, 2014, pp. 397–412.

[109] Terry R Payne and Valentina Tamma, *Negotiating over ontological correspondences with asymmetric and incomplete knowledge*, Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems, International Foundation for Autonomous Agents and Multiagent Systems, 2014, pp. 517–524.

[110] Terry R Payne and Valentina Tamma, *Using Preferences in Negotiations over Ontological Correspondences*, International Conference on Principles and Practice of Multi-Agent Systems, Springer International Publishing, 2015, pp. 319–334.

[111] Oxford University Press, *Oxford English Dictionary*, `http://www.oed.com/view/Entry/131551?redirectedFrom=ontology#eid` (visited on 17 September 2016).

[112] Dominique Ritze, Christian Meilicke, Ondřej Šváb-Zamazal, and Heiner Stuckenschmidt, *A pattern-based ontology matching approach for detecting complex correspondences*, Proceedings of the 4th International Conference on Ontology Matching-Volume 551, CEUR-WS. org, 2009, pp. 25–36.

[113] Dominique Ritze, Johanna Völker, Christian Meilicke, and Ondrej Šváb-Zamazal, *Linguistic analysis for complex ontology matching*, Proceedings of the 5th International Conference on Ontology Matching, vol. 689, CEUR-WS. org, 2010, pp. 1 –12.

[114] Cornelius Rosse and José LV Mejino Jr, *The foundational model of anatomy ontology*, Anatomy Ontologies for Bioinformatics, Springer, 2008, pp. 59–117.

[115] James Rumbaugh, Ivar Jacobson, and Grady Booch, *Unified modeling language reference manual, the*, Pearson Higher Education, 2004.

[116] Gabrielle Santos, Valentina Tamma, Terry R Payne, and Floriana Grasso, *Dialogue Based Meaning Negotiation*, The 15th Workshop on Computational Models of Natural Argument (CMNA 2015), 2015.

[117] Gabrielle Santos, Valentina Tamma, Terry R Payne, and Floriana Grasso, *A Dialogue Protocol to Support Meaning Negotiation*, The 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS2016), 2016.

[118] Ulrike Sattler, Thomas Schneider, and Michael Zakharyaschev, *Which kind of module should I extract?*, Description Logics **477** (2009), 78.

[119] François Scharffe, Jérome Euzenat, Ying Ding, and Dieter Fensel, *Correspondence patterns for ontology mediation*, Proceedings of the Ontology Matching Workshop at ISWC, 2007.

[120] François Scharffe and Dieter Fensel, *Correspondence patterns for ontology alignment*, International Conference on Knowledge Engineering and Knowledge Management, Springer, 2008, pp. 83–92.

[121] François Scharffe, Ondřej Zamazal, and Dieter Fensel, *Ontology alignment design patterns*, Knowledge and information systems **40** (2014), no. 1, 1–28.

[122] Luciano Serafini and Andrei Tamilin, *Drago: Distributed reasoning architecture for the semantic web*, European Semantic Web Conference, Springer, 2005, pp. 361–376.

[123] Inanç Seylan, Enrico Franconi, and Jos De Bruijn, *Effective query rewriting with ontologies over DBoxes*, IJCAI, vol. 9, Citeseer, 2009, pp. 923–929.

[124] Pavel Shvaiko and Jérôme Euzenat, *Ontology matching: state of the art and future challenges*, Knowledge and Data Engineering, IEEE Transactions on **25** (2013), no. 1, 158–176.

[125] Nuno Silva, Paulo Maio, and João Rocha, *An Approach to Ontology Mapping Negotiation.*, Integrating Ontologies, 2005.

[126] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz, *Pellet: A practical OWL-DL reasoner*, Web Semantics: science, services and agents on the World Wide Web **5** (2007), no. 2, 51–53.

[127] Alessandro Solimando, Ernesto Jiménez-Ruiz, and Giovanna Guerrini, *Detecting and correcting conservativity principle violations in ontology-to-ontology mappings*, The Semantic Web–ISWC 2014, Springer, 2014, pp. 1–16.

[128] Vassilis Spiliopoulos and George A Vouros, *Synthesizing ontology alignment methods using the max-sum algorithm*, IEEE Transactions on Knowledge and Data Engineering **24** (2012), no. 5, 940–951.

[129] Robert Stevens, *Being complex on the left-hand-side: General Concept Inclusions*, `http://ontogenesis.knowledgeblog.org/1288` (visited on 22 September 2016).

[130] Heiner Stuckenschmidt, Christine Parent, and Stefano Spaccapietra, *Modular ontologies: concepts, theories and techniques for knowledge modularization*, vol. 5445, Springer, 2009.

[131] Heiner Stuckenschmidt, Livia Predoiu, and Christian Meilicke, *Learning Complex Ontology Alignments A Challenge for ILP Research*, Proceedings of the 18th International Conference on Inductive Logic Programming, 2008.

[132] Rudi Studer, V Richard Benjamins, and Dieter Fensel, *Knowledge engineering: principles and methods*, Data & knowledge engineering **25** (1998), no. 1, 161–197.

[133] Ondřej Šváb-Zamazal, Vojtěch Svátek, and Luigi Iannone, *Pattern-based ontology transformation service exploiting OPPL and OWL-API*, Knowledge Engineering and Management by the Masses, Springer, 2010, pp. 105–119.

[134] Balder Ten Cate, Willem Conradie, Maarten Marx, and Yde Venema, *Definitorially Complete Description Logics*, KR **6** (2006), 79–89.

[135] Balder Ten Cate, Enrico Franconi, and Inanç Seylan, *Beth Definability in Expressive Description Logics*, Journal of Artificial Intelligence Research (JAIR) **48** (2013), 347–414.

[136] Balder ten Cate, Enrico Franconi, and İnanç Seylan, *Beth Definability in Expressive Description Logics*, Journal of Artificial Intelligence Research **48** (2013), no. 1, 347–414.

[137] Cássia Trojahn, Jérôme Euzenat, Valentina Tamma, and Terry R Payne, *Argumentation for reconciling agent ontologies*, Semantic Agent Systems, Springer, 2011, pp. 89–111.

[138] Cássia Trojahn, Márcia Moraes, Paulo Quaresma, and Renata Vieira, *A cooperative approach for composite ontology mapping*, Journal on data semantics X, Springer, 2008, pp. 237–263.

[139] Cassia Trojahn, Paulo Quaresma, and Renata Vieira, *Conjunctive queries for ontology based agent communication in MAS*, Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2, International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 829–836.

[140] Dmitry Tsarkov and Ian Horrocks, *FaCT++ Description Logic reasoner: System description*, Automated reasoning, Springer, 2006, pp. 292–297.

[141] Mike Uschold and Michael Gruninger, *Ontologies: Principles, methods and applications*, The knowledge engineering review **11** (1996), no. 02, 93–136.

[142] Frank Van Harmelen, Annette Ten Teije, and Holger Wache, *Knowledge engineering rediscovered: Towards reasoning patterns for the semantic web*, Foundations for the Web of Information and Services, Springer, 2011, pp. 57–75.

[143] Gertjan Van Heijst, A Th Schreiber, and Bob J Wielinga, *Using explicit ontologies in KBS development*, International journal of human-computer studies **46** (1997), no. 2, 183–292.

[144] Moshe Y Vardi, *Fundamentals of dependency theory*, IBM Thomas J. Watson Research Division, 1985.

[145] Vijay V Vazirani, *Approximation algorithms*, Springer Science & Business Media, 2013.

[146] P.R.S. Visser, D.M. Jones, T.J.M. Bench-Capon, and M.J.R. Shave, *Assessing heterogeneity by classifying ontology mismatches*, Formal Ontology in Information Systems. Proceedings FOIS'98, Trento, Italy (N. Guarino, ed.), IOS Press, 1998, pp. 148–182.

[147] W3C, *World Wide Web Consortium (W3C) About the Consortium*, June 2016, `https://www.w3.org/Consortium/` (visited on 10 June 2016).

[148] Brian Walshe, Rob Brennan, and Declan O'Sullivan, *A comparison of complex correspondence detection techniques*, Proceedings of the 7th International Conference on Ontology Matching, vol. 946, CEUR-WS.org, 2012, pp. 236–237.

[149] Alfred North Whitehead and Bertrand Russell, *Principia mathematica*, vol. 2, University Press, 1912.

[150] Michael Wooldridge, *An Introduction to Multiagent Systems*, John Wiley & Sons, 2009.

[151] Michael Wooldridge and Nicholas R Jennings, *Intelligent agents: Theory and practice*, The Knowledge Engineering Review **10** (1995), no. 02, 115–152.

[152] Ondřej Zamazal and Vojtěch Svátek, *The ten-year ontofarm and its fertilization within the onto-sphere*, Web Semantics: Science, Services and Agents on the World Wide Web **43** (2017), no. 1.